Distributed Signal Processing Algorithms for Wireless Networks

Songcen Xu

Doctor of Philosophy

University of York

Electronics

July 2015

Abstract

Distributed signal processing algorithms have become a key approach for statistical inference in wireless networks and applications such as wireless sensor networks and smart grids. It is well known that distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit the estimates to a specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved estimate. In this thesis, novel distributed cooperative algorithms for inference in ad hoc, wireless sensor networks and smart grids are investigated. Low-complexity and effective algorithms to perform statistical inference in a distributed way are devised. A number of innovative approaches for dealing with node failures, compression of data and exchange of information are proposed and summarized as follows: Firstly, distributed adaptive algorithms based on the conjugate gradient (CG) method for distributed networks are presented. Both incremental and diffusion adaptive solutions are considered. Secondly, adaptive link selection algorithms for distributed estimation and their application to wireless sensor networks and smart grids are proposed. Thirdly, a novel distributed compressed estimation scheme is introduced for sparse signals and systems based on compressive sensing techniques. The proposed scheme consists of compression and decompression modules inspired by compressive sensing to perform distributed compressed estimation. A design procedure is also presented and an algorithm is developed to optimize measurement matrices. Lastly, a novel distributed reduced-rank scheme and adaptive algorithms are proposed for distributed estimation in wireless sensor networks and smart grids. The proposed distributed scheme is based on a transformation that performs dimensionality reduction at each agent of the network followed by a reduced–dimension parameter vector.

Contents

Abstract ii

List of Figures ix

List of Tables xiii

Acknowledgements xv

Declaration xvi

1 Introduction 1

1.1 Overview 1

1.2 Motivation 3

1.3 Contributions 4

1.4 Thesis Outline 5

1.5 Notation 7

2 Literature Review 8

2.1 Distributed Signal Processing 8
2.1.1 Distributed Wireless Networks 9
2.1.2 Applications 10
2.2 Protocols for Cooperation and Exchange of Information 15
2.2.1 Incremental Strategy 15
2.2.2 Diffusion Strategy 16
2.2.3 Consensus Strategy 19
2.3 Adaptive Algorithms 20
2.3.1 The Least Mean Square (LMS) Algorithm 22
2.3.2 The Recursive Least Squares (RLS) Algorithm 23
2.3.3 The Conjugate Gradient (CG) Algorithm 24
2.4 Compressive Sensing Techniques 25
2.4.1 Goal of Compressive Sensing 26
2.4.2 Measurement Matrix 27
2.4.3 Reconstruction Algorithms 28
2.5 Sparsity–Aware Techniques 30
2.5.1 The Zero–Attracting Strategy 31
2.5.2 The Reweighted Zero–Attracting Strategy 32
2.5.3 Simulation results 32

2.6 Summary 34

3 Distributed Conjugate Gradient Strategies for Distributed Estimation Over Sensor Networks 36

3.1 Introduction 36

3.2 System Models 38

3.2.1 Distributed Parameter Estimation 38

3.2.2 Distributed Spectrum Estimation 39

3.3 Proposed Incremental Distributed CG–Based Algorithms 41

3.3.1 Incremental Distributed CG–Based Solutions 41

3.3.2 Computational Complexity 46

3.4 Proposed Diffusion Distributed CG–Based Algorithms 46

3.4.1 Diffusion Distributed CG–Based Algorithms 47

3.4.2 Computational Complexity 48

3.5 Preconditioner Design 48

3.6 Simulation Results 52

3.6.1 Distributed Estimation in Wireless Sensor Networks 52

3.6.2 Distributed Spectrum Estimation 56

3.7 Summary 58

4 Adaptive Link Selection Algorithms for Distributed Diffusion Estimation 59

4.1 Introduction . . . 59

4.1.1 Prior and Related Work . . . 60

4.1.2 Contributions . . . 60

4.2 System Model and Problem Statement . . . 62

4.3 Proposed Adaptive Link Selection Algorithms . . . 64

4.3.1 Exhaustive Search–Based LMS/RLS Link Selection . . . 65

4.3.2 Sparsity–Inspired LMS/RLS Link Selection . . . 67

4.4 Analysis of the proposed algorithms . . . 72

4.4.1 Stability Analysis . . . 74

4.4.2 MSE Steady–State Analysis . . . 77

4.4.3 Tracking Analysis . . . 87

4.4.4 Computational Complexity . . . 89

4.5 Simulations . . . 90

4.5.1 Diffusion Wireless Sensor Networks . . . 91

4.5.2 MSE Analytical Results . . . 95

4.5.3 Smart Grids . . . 96

4.6 Summary . . . 100

5 Distributed Compressed Estimation Based on Compressive Sensing 101

5.1 Introduction 101

5.2 System Model and Problem Statement 103

5.3 Proposed Distributed Compressed Estimation Scheme 104

5.4 Measurement Matrix Optimization 108

5.5 Simulations 110

5.6 Summary 113

6 Distributed Reduced–Rank Estimation Based on Joint Iterative Optimization in Sensor Networks 114

6.1 Introduction 114

6.2 System Model 116

6.3 Distributed Dimensionality Reduction and Adaptive Processing 118

6.4 Proposed Distributed Reduced-Rank Algorithms 119

6.4.1 Proposed DRJIO–NLMS algorithm 120

6.4.2 Proposed DRJIO–RLS algorithm 123

6.4.3 Computational Complexity Analysis 124

6.5 Simulation results 126

6.5.1 Wireless Sensor Networks 127

6.5.2 Smart Grids 130

6.6 Summary 134

7 Conclusions and Future Work **135**

7.1 Summary of the Work 135

7.2 Future Work 137

Glossary **139**

Bibliography **141**

List of Figures

1.1 Distributed wireless network sample 2

2.1 Example of a network topology with N=7 nodes 11

2.2 Example of the IEEE 14–bus system 14

2.3 Incremental Distributed Estimation 16

2.4 Diffusion Distributed Estimation 18

2.5 Consensus Distributed Estimation 20

2.6 Adaptive Filter Structure 21

2.7 System identification 33

2.8 MSD comparison for scenario 1 34

2.9 MSD comparison for scenario 2 35

2.10 MSD comparison for scenario 3 35

3.1 Incremental distributed CG–based network processing 41

3.2 Diffusion Distributed CG–Based Network Processing 47

3.3 MSD performance comparison for the incremental distributed strategies 53

3.4 MSE performance comparison for the incremental distributed strategies 54

3.5 Network structure 55

3.6 MSD performance comparison for the diffusion distributed strategies 55

3.7 MSE performance comparison for the diffusion distributed strategies 56

3.8 Performance comparison for the distributed spectrum estimation 57

3.9 Example of distributed spectrum estimation 58

4.1 Network topology with N nodes 62

4.2 Sparsity aware signal processing strategies 69

4.3 Covariance matrices $\boldsymbol{A}_3$ for different sets of Ω_3 80

4.4 Covariance matrix $\boldsymbol{A}_3$ upon convergence 83

4.5 Diffusion wireless sensor networks topology with 20 nodes 91

4.6 Network MSE curves in a static scenario 93

4.7 Network MSE curves in a time-varying scenario 93

4.8 Set of selected links for node 16 with ES–LMS in a static scenario 94

4.9 Link selection state for node 16 with ES–LMS in a time-varying scenario 94

4.10 MSE steady–state performance against SNR for ES–LMS and SI–LMS 95

4.11 MSE steady–state performance against SNR for ES–RLS and SI–RLS 96

4.12 MSE steady–state performance against SNR for ES–LMS and SI–LMS in a time-varying scenario . . . 97

4.13 MSE steady–state performance against SNR for ES–RLS and SI–RLS in a time-varying scenario . . . 97

4.14 IEEE 14–bus system for simulation . . . 99

4.15 MSE performance curves for smart grids . . . 99

5.1 Proposed Compressive Sensing Modules . . . 105

5.2 Proposed DCE Scheme . . . 105

5.3 Diffusion wireless sensor network with 20 Nodes . . . 110

5.4 MSE performance against time . . . 112

5.5 MSE performance against reduced dimension D for different levels of resolution in bits per coefficient . . . 112

6.1 Network topology with 20 nodes . . . 117

6.2 Proposed dimensionality reduction scheme at each node or agent . . . 118

6.3 Complexity in terms of multiplications . . . 126

6.4 Full–rank system with M=20 . . . 128

6.5 Sparse system with M=20 . . . 128

6.6 Full–rank system with M=60 . . . 129

6.7 DRJIO–NLMS vs DCE scheme with sparsity level S=3 . . . 131

6.8 DRJIO–NLMS vs DCE scheme with sparsity level S=10 131

6.9 Hierarchical IEEE 14–bus system 133

6.10 MSE performance for smart grids 133

List of Tables

2.1 Diffusion LMS ATC strategy 17

2.2 Diffusion LMS CTA strategy 18

2.3 Comparison of Main Steps for ATC and CTA strategies 19

2.4 Main Steps for CG algorithm 25

3.1 IDCCG Algorithm 43

3.2 IDMCG Algorithm 45

3.3 Computational Complexity of Different Incremental Algorithms per Node 46

3.4 DDCCG Algorithm 49

3.5 DDMCG Algorithm 50

3.6 Computational Complexity Of Different Diffusion Algorithms per Node . 50

4.1 The ES-LMS Algorithm 67

4.2 The ES-RLS Algorithm 68

4.3 The SI-LMS and SI-RLS Algorithms 73

4.4 Coefficients α_{kl} for different sets of Ω_3 80

4.5 Computational complexity for the adaptation step per node per time instant 90

4.6 Computational complexity for combination step per node per time instant 90

4.7 Computational complexity per node per time instant 90

5.1 The Proposed DCE Scheme 107

6.1 The DRJIO–NLMS Algorithm 122

6.2 The DRJIO-RLS Algorithm 125

6.3 Computational Complexity of Different Algorithms per Node 127

Acknowledgements

I would like to show my sincere gratitude to my supervisor, Prof. Rodrigo C. de Lamare, for his support and encouragement during my PhD study.

I am very grateful to my thesis advisor, Dr. Yuriy Zakharov, whose insightful discussions and suggestions have benefited me.

Special thanks are also extended to Prof. Vincent Poor from Princeton University for the collaboration which resulted in several papers.

I would also like to thank Dr. Peng Tong, Dr. Yi Wang, and Dr. Dong Fang and other colleagues in the Communications and Signal Processing Research Group.

This thesis is dedicated to my parents and my love Jiaqi Gu.

Declaration

All work presented in this thesis as original is so, to the best knowledge of the author. References and acknowledgements to other researchers have been given as appropriate.

Elements of the research presented in this thesis have resulted in some publications. A list of these publications can be found below.

Journal Papers

1. S. Xu, R. C. de Lamare and H. V. Poor, "Distributed ReducedRank Adaptive Estimation Based on Alternating Optimization", submitted to *IEEE Transactions on Signal Processing.*
2. S. Xu, R. C. de Lamare and H. V. Poor, "Distributed Estimation Over Sensor Networks Based on Distributed Conjugate Gradient Strategies", submitted to *IET Signal Processing.*
3. S. Xu, R. C. de Lamare and H. V. Poor, "Adaptive Link Selection Algorithms for Distributed Estimation", submitted to *EURASIP J Adv Signal Process.*
4. S. Xu, R. C. de Lamare and H. V. Poor, "Distributed Compressed Estimation Based on Compressive Sensing", *IEEE Signal Processing Letters.*, vol. 22, no. 9, pp. 1311-1315, September. 2015.

Conference Papers

1. T. G. Miller, S. Xu and R. C. de Lamare, "Consensus Distributed Conjugate Gradient Algorithms for Parameter Estimation over Sensor Networks", Accepted, *XXXIII Brazilian Telecommunications Symposium*, Juiz de Fora, Brazil, 2015.

2. S. Xu, R. C. de Lamare and H. V. Poor, "Distributed Reduced-Rank Estimation Based on Joint Iterative Optimization in Sensor Networks", *European Signal. Processing Conference*, pp. 2360-2364, Lisbon, Portugal, September 2014.

3. S. Xu, R. C. de Lamare and H. V. Poor, "Dynamic Topology Adaptation for Distributed Estimation in Smart Grids", *Proc. IEEE workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 420-423, Saint Martin, December 2013.

4. S. Xu, R. C. de Lamare and H. V. Poor, "Adaptive Link Selection Strategies for Distributed Estimation in Diffusion Wireless Networks", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5402-5405, Vancouver, Canada, May 2013.

5. S. Xu and R. C. de Lamare, "Distributed Conjugate Gradient Strategies for Distributed Estimation over Sensor Networks", *Proc. Sensor Signal Processing for Defence* pp. 1-5, London, UK, September 2012.

Chapter 1

Introduction

Contents

1.1 Overview 1

1.2 Motivation 3

1.3 Contributions 4

1.4 Thesis Outline 5

1.5 Notation 7

1.1 Overview

Distributed signal processing algorithms have become of paramount importance for statistical inference in wireless networks and applications such as wireless sensor networks, spectrum estimation and smart grids [1–4]. It is well known that distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area [1]. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit their estimates to the specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved estimate. When compared with the centralized solution, the distributed solution has significant advantages. The centralized solution needs a central processor, where each node sends its information to the central processor and gets

the information back after the processor completes the task. This type of communication needs the central processor to be powerful and reliable enough. With distributed solutions, each node only requires the local information and its neighbors to process the information. This approach for processing information can significantly reduce the amount of processing and the communications bandwidth requirements. In Fig. 1.1, the idea of distributed signal processing is illustrated, where each node stands for a sensor and the rectangle corresponds to the target to estimate. In detail, the aim of the distributed network is to estimate the unknown target with the help of the cooperation between sensors.

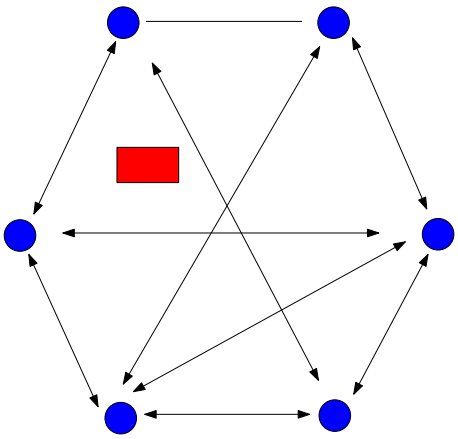

Figure 1.1: Distributed wireless network sample

There are three main protocols for cooperation and exchange of information for distributed processing, incremental, diffusion and consensus strategies, and recent studies indicate that the diffusion strategy is the most effective one [5]. Details of each strategy will be introduced and discussed in Chapter 2. For distributed diffusion processing, many challenges still exist. Firstly, the neighbors for each node are fixed and the combining coefficients are calculated after the network topology is deployed and starts its operation. One disadvantage of this approach is that the estimation procedure may be affected by poorly performing links. Moreover, when the number of neighbor nodes is large, the demand of bandwidth to transmit the estimate between neighbour nodes is high. Secondly, in many scenarios, when the unknown parameter vector to be estimated has a large dimension, the network requires a large communication bandwidth between neighbor nodes to transmit their local estimate. This problem limits the application of existing algorithms in applications with large data sets. Meanwhile, as the convergence speed is dependent on the length of the parameter vector, the large dimension also becomes a challenge for

existing algorithms. Thirdly, in some scenarios, the unknown parameter vector to be estimated can be sparse and contain only a few nonzero coefficients. However, when the full dimension of the observed data is taken into account, the challenge lies in the increased computational cost, the slowed down convergence rate and the degraded mean square error (MSE) performance.

1.2 Motivation

In this thesis, a number of innovative distributed cooperative strategies for dealing with exchange of information, node failures and compression of data are considered for the applications of wireless sensor networks, spectrum estimation and smart grids, which require low complexity and are highly effective to perform statistical inference about the environment in a distributed way. Firstly, the conjugate gradient (CG) method [6–8] is considered to design distributed adaptive algorithms for distributed networks. In particular, both incremental and diffusion adaptive solutions are proposed. The distributed conventional CG (CCG) and modified CG (MCG) algorithms provide an improved performance in terms of MSE as compared with least–mean–square (LMS)–based algorithms and a performance that is close to recursive least-squares (RLS) algorithms. The proposed distributed CG algorithms are examined for parameter estimation in wireless sensor networks and spectrum estimation.

Secondly, adaptive link selection algorithms for distributed estimation and their application to wireless sensor networks and smart grids are investigated. Specifically, based on the LMS/RLS strategies [9, 10], exhaustive search-based LMS/RLS link selection algorithms and sparsity-inspired LMS/RLS link selection algorithms that can exploit the topology of networks with poor-quality links are considered. The proposed link selection algorithms are then analyzed in terms of their stability, steady-state and tracking performance, and computational complexity.

Thirdly, a distributed compressed estimation scheme is proposed for sparse systems based on compressive sensing techniques [11, 12]. Inspired by compressive sensing, the proposed scheme consists of compression and decompression modules to perform dis-

tributed compressed estimation. To further improve the performance of the proposed distributed compressed estimation scheme, an algorithm is developed to optimize measurement matrices. The proposed distributed compressed estimation scheme and algorithms are assessed for parameter estimation problems in wireless sensor networks.

Fourthly, the challenge that estimating large dimension unknown parameter vectors in wireless sensor networks requires large communication bandwidth is addressed. In particular, we develop a novel distributed reduced-rank scheme and adaptive algorithms for performing distributed dimensionality reduction and computing low–rank approximations of unknown parameter vectors. The proposed distributed scheme is based on a transformation that performs dimensionality reduction at each agent of the network followed by a reduced-dimension parameter vector [13]. Distributed reduced–rank joint iterative estimation algorithms based on the NLMS and RLS techniques are also developed to achieve significantly reduced communication overhead and improved performance.

1.3 Contributions

The contributions presented in this thesis are summarised as follows:

- Based on the fact that the CG algorithm has a faster convergence rate [6] than the steepest descent algorithms and a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG algorithm employs [8], two CG-based incremental distributed solutions and two diffusion distributed CG-based strategies are proposed. In detail, the incremental distributed CCG solution (IDCCG), incremental distributed MCG solution (IDMCG), diffusion distributed CCG solution (DDCCG) and diffusion distributed MCG solution (DDMCG) are developed and analysed in terms of their computational complexity. These algorithms can be used in the applications, such as distributed estimation and spectrum estimation.

- The adaptive link selection algorithms for distributed estimation problems are proposed and studied. Specifically, we develop adaptive link selection algorithms that can exploit the knowledge of poor links by selecting a subset of data from neighbor

nodes. The first approach consists of exhaustive search–based LMS/RLS link selection (ES–LMS/ES–RLS) algorithms, whereas the second technique is based on sparsity–inspired LMS/RLS link selection (SI–LMS/SI–RLS) algorithms. The proposed algorithms result in improved estimation performance in terms of the MSE associated with the estimates. In contrast to previously reported techniques, a key feature of the proposed algorithms is that they involve only a subset of the data associated with the best performing links.

- We propose the design of a scheme that can exploit lower dimensions and the sparsity present in the signals, reduce the required bandwidth, and improve the convergence rate and the MSE performance. Inspired by CS, the scheme that incorporates compression and decompression modules into the distributed estimation procedure is introduced and namely distributed compressed estimation (DCE) scheme. We also present a design procedure and develop an algorithm to optimize the measurement matrices, which can further improve the performance of the proposed scheme. Specifically, we derive an adaptive stochastic gradient recursion to update the measurement matrix.

- A scheme for distributed signal processing along with distributed reduced–rank algorithms for parameter estimation is presented. In particular, the proposed algorithms are based on an alternating optimization strategy and are called distributed reduced-rank joint iterative optimization normalized least mean squares (DRJIO–NLMS) algorithm and distributed reduced–rank joint iterative optimization recursive least square (DRJIO–RLS) algorithm. In contrast to prior work on reduced–rank techniques and distributed methods, the proposed reduced–rank strategies are distributed and perform dimensionality reduction without costly decompositions at each agent. The proposed DRJIO–NLMS and DRJIO–RLS algorithms are flexible with regards to the amount of information that is exchanged, have low cost and high performance.

1.4 Thesis Outline

The structure of the thesis is as follows:

- Chapter 2 presents an overview of the theory relevant to this thesis and introduces the applications that are used to present the proposed algorithms and schemes in this thesis. The topics of distributed signal processing, incremental and diffusion strategies, adaptive signal processing, compressive sensing and sparsity–aware techniques are covered with an outline of previous work in these fields and important applications.

- Chapter 3 presents the design of distributed adaptive algorithms for distributed networks based on CG strategies. Both incremental and diffusion adaptive solutions are proposed, alongside a computational complexity analysis and the application to distributed estimation and spectrum estimation.

- In Chapter 4, adaptive link selection algorithms for distributed estimation and their application to wireless sensor networks and smart grids are proposed. The analysis of the proposed algorithms are presented in terms of their stability, steady-state and tracking performance, and computational complexity.

- Chapter 5 presents a novel distributed compressed estimation scheme for sparse signals and systems based on compressive sensing techniques. The compression and decompression modules inspired by compressive sensing are introduced to perform distributed compressed estimation. A design procedure to optimize measurement matrices is presented as well. The simulation results are also presented in a wireless sensor networks application to show that the DCE scheme outperforms existing strategies in terms of convergence rate, reduced bandwidth and MSE performance.

- Chapter 6 presents a novel distributed reduced-rank scheme and adaptive algorithms for distributed estimation in wireless sensor networks and smart grids. A distributed reduced–rank joint iterative estimation algorithm is developed and compared with existing techniques. Simulation results are provided to illustrator that the proposed algorithms have better performance than existing algorithms. We have also compared the proposed algorithms with the DCE scheme, which was presented in chapter 5, for systems with different levels of sparsity.

- Chapter 7 presents the conclusions of this thesis, and suggests directions in which further research could be carried out.

1.5 Notation

$\boldsymbol{a}$	the vector (boldface lower case letters)
$\boldsymbol{A}$	the matrix (boldface upper case letters)
$\Re$	real part
$\boldsymbol{I}_N$	$N \times N$ identity matrix
$(\cdot)^*$	complex conjugate
$(\cdot)^T$	matrix transpose
$(\cdot)^H$	Hermitian transpose
$\mathbb{E}$	expectation operator
$\otimes$	Kronecker product
$\mathrm{tr}(\cdot)$	trace of a matrix
$\langle \cdot, \cdot \rangle$	inner product

Chapter 2

Literature Review

Contents

2.1 Distributed Signal Processing 8

2.2 Protocols for Cooperation and Exchange of Information 15

2.3 Adaptive Algorithms 20

2.4 Compressive Sensing Techniques 25

2.5 Sparsity–Aware Techniques 30

2.6 Summary 34

In this chapter, an introduction to fundamental techniques related to the research carried out during the preparation of this thesis such as distributed signal processing, protocols for cooperation and exchange of information, adaptive algorithms, compressive sensing and sparsity–aware techniques are presented.

2.1 Distributed Signal Processing

Distributed signal processing deals with information processing over graphs and the method of collaboration among nodes in a network, and aims to provide a superior adaptation performance [14]. Distributed signal processing covers strategies and results that relate to the design and analysis of networks that are able to solve optimization and adap-

tation problems in an efficient and distributed manner. One main research aspect for distributed processing is how to perform distributed adaptation and optimization over networks. Under this topic, the advantages and limitations of centralized and distributed solutions are examined and compared.

In the centralized mode of operation, each node transmits their data to a fusion center, which has the function of processing the data centrally. The fusion center then transmits the results of the analysis back to the nodes. Although centralized solutions sometimes can be powerful, they may suffer from some limitations. In real–time applications where nodes collect data continuously, the reiterant exchange of information between the nodes and the fusion center can be costly especially when these exchanges occur over wireless networks or limited communication bandwidth. In addition, centralized solutions may face a critical failure. When a central processor fails, this solution method collapses altogether [14].

On the other hand, in the distributed mode of operation, nodes are connected by a topology and share information only with their immediate neighbors. The continuous diffusion of information across the network enables nodes to adapt their performance in relation to data and network conditions. It also results in improved adaptation performance relative to non–cooperative nodes [15]. In addition, the distributed solution has recently been introduced for parameter estimation in wireless networks and power networks [1–4], which show the advantages of distributed solution over centralized solutions.

2.1.1 Distributed Wireless Networks

A distributed wireless network is one topic of distributed signal processing. Distributed wireless networks linking PCs, laptops, cell phones, sensors and actuators will form the backbone of future data communication and control networks [5]. Distributed networks have their own characteristics; any node in the network will be connected with at least two other nodes directly, which means the network has increased reliability. For example, compared with centrally controlled networks, since the distributed network does not contain a central controller, the whole network will not crash when the center is under attack. Moreover, in a distributed network the nodes are connected to each other. This enables

the network with the ability to choose multiple routes for data transmission. Furthermore, it can help to reduce the negative effect of fading, noise, interference and so on.

Distributed wireless networks deal with the extraction of information from data collected at nodes that are distributed over a geographic area [1]. In distributed wireless networks, each node will collect information about the target in the network and exchange this information with other nodes according to the network topology, and finally produce an estimate of the parameters of interest. Several algorithms have already been developed and successfully implemented for distributed wireless networks, i.e, steepest–descent, LMS [1], affine projection (AP) [16] and RLS [17, 18], which are reported in the literature. In the following chapters, we compare our proposed algorithms with some of these existing algorithms.

2.1.2 Applications

In this subsection, the applications of distributed signal processing are presented. We focus on three main applications, which are distributed estimation, spectrum estimation and smart grids.

Distributed Estimation

In general, for distributed estimation, a set of nodes is required to collectively estimate some parameter of interest from noisy measurements. Thus, consider a set of N nodes, which have limited processing capabilities, distributed over a given geographical area as depicted in Fig. 2.1. In this scenario, limited processing capabilities means the nodes are connected and form a network, which is assumed to be partially connected because nodes can exchange information only with neighbors determined by the connectivity topology. We call a network with this property a partially connected network whereas a fully connected network means that data broadcast by a node can be captured by all other nodes in the network in one hop [19].

The aim of distributed estimation is to estimate an unknown parameter vector $\boldsymbol{\omega}_0$,

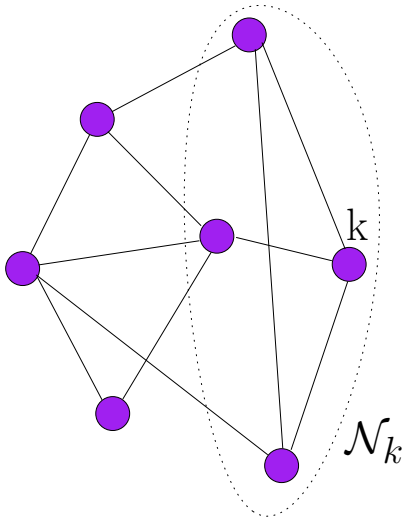

Figure 2.1: Example of a network topology with N=7 nodes

which has length M. At every time instant i, each node k takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{2.1}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ random regression input signal vector and $n_k(i)$ denotes the Gaussian noise at each node with zero mean and variance $\sigma_{n,k}^2$. This linear model is able to capture or approximate well many input–output relations for estimation purposes [14] and we assume $I > M$. To compute an estimate of $\boldsymbol{\omega_0}$ in a distributed fashion, we need each node to minimize the MSE cost function [2]

$$J_k\big(\boldsymbol{\omega}_k(i)\big) = \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2, \tag{2.2}$$

where $\mathbb{E}$ denotes expectation and $\boldsymbol{\omega}_k(i)$ is the estimated vector generated by node k at time instant i. Then, the global network cost function could be described as

$$J\big(\boldsymbol{\omega}\big) = \sum_{k=1}^{N} \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}^H \boldsymbol{x}_k(i)\big|^2. \tag{2.3}$$

Many distributed estimation algorithms have been proposed to minimize the cost function (2.3), in the context of distributed adaptive filtering. These include incremental LMS [1, 5], incremental RLS [5], diffusion LMS [2, 5] and diffusion RLS [20]. Kalman filtering and smoothing algorithms were also proposed in [21].

Spectrum Estimation

In this subsection, the application of distributed signal processing techniques for spectrum estimation is introduced. We aim to estimate the spectrum of a transmitted signal $\boldsymbol{s}$ using a wireless sensor network with N nodes. For example, in a cognitive network, distributed spectrum estimation could be used to sense the spectrum before allocating a radio resources. Let $\boldsymbol{\Phi}_s(f)$ denote the power spectral density (PSD) of the signal $\boldsymbol{s}$. The PSD can be represented as a linear combination of some $\mathcal{B}$ basis functions, as described by

$$\Phi_s(f) = \sum_{m=1}^{\mathcal{B}} b_m(f)\omega_{0m} = \boldsymbol{b}_0^T(f)\boldsymbol{\omega}_0, \tag{2.4}$$

where $\boldsymbol{b}_0(f) = [b_1(f), ..., b_{\mathcal{B}}(f)]^T$ is the vector of basis functions evaluated at frequency f, $\boldsymbol{\omega}_0 = [\omega_{01}, ..., \omega_{0\mathcal{B}}]$ denote the expansion coefficients to be estimated, and $\mathcal{B}$ is the number of basis functions. For $\mathcal{B}$ sufficiently large, the basis expansion in (2.4) can well approximate the transmitted spectrum. Possible choices for the set of basis $\{b_m(f)\}_{m=1}^{\mathcal{B}}$ include [22–24]:

- Rectangular functions
- Raised cosines
- Gaussian bells
- Splines

Let $H_k(f, i)$ be the channel transfer function between a transmit source conveying the signal $\boldsymbol{s}$ and receive node k at time instant i, the PSD of the received signal observed by node k can be expressed as

$$\begin{aligned} I_k(f, i) &= |H_k(f, i)|^2\Phi_s(f) + v_k^2 \\ &= \sum_{m=1}^{\mathcal{B}} |H_k(f, i)|^2 b_m(f)\omega_{0m} + v_k^2 \\ &= \boldsymbol{b}_{k,i}^T(f)\boldsymbol{\omega}_0 + v_k^2 \end{aligned} \tag{2.5}$$

where $\boldsymbol{b}_{k,i}^T(f) = [|H_k(f, i)|^2 b_m(f)]_{m=1}^{\mathcal{B}}$ and v_k^2 is the receiver noise power at node k. For simplification, let us assume that the link between receive node k and the transmit source is perfect.

At every time instance i, every node k observes measurements of the noisy version of the true PSD $I_k(f,i)$ described by (2.5) over N_c frequency samples $f_j = f_{min} : (f_{max} - f_{min})/N_c : f_{max}$, for $j = 1, ..., N_c$, according to the model:

$$d_k^j(i) = \boldsymbol{b}_{k,i}^T(f_j)\boldsymbol{\omega}_0 + v_k^2 + n_k^j(i). \tag{2.6}$$

The term $n_k^j(i)$ denotes sampling noise and is assumed to have zero mean and variance $\sigma_{n,j}^2$. The receiver noise power v_k^2 can be estimated with high accuracy in a preliminary step using, e.g., an energy estimator over an idle band, and then subtracted from (2.6) [25, 26]. Thus, collecting measurements over N_c contiguous frequencies, we obtain a linear model given by

$$\boldsymbol{d}_k(i) = \boldsymbol{B}_k(i)\boldsymbol{\omega}_0 + \boldsymbol{n}_k(i), \tag{2.7}$$

where $\boldsymbol{B}_k(i) = [\boldsymbol{b}_{k,i}^T(f_j)]_{j=1}^{N_c} \in \mathbb{R}^{N_c \times \mathcal{B}}$, with $N_c > \mathcal{B}$ and $\boldsymbol{n}_k(i) = [n_k^1(i), ...n_k^{N_c}(i)]^T$. At this point, we can generate the cost function for node k as:

$$J_k(\boldsymbol{\omega}_k(i)) = \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}_k(i)\big|^2 \tag{2.8}$$

and the global network cost function could be described as

$$J\big(\boldsymbol{\omega}_k(i)\big) = \sum_{k=1}^{N} \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}_k(i)\big|^2. \tag{2.9}$$

In the literature, some distributed estimation algorithms have been proposed to minimize the cost function in (2.9), including cooperative sparse PSD estimation [22] and sparse distributed spectrum estimation [25].

Smart Grids

The electric power industry is likely to involve many more fast information gathering and processing devices (e.g., phasor measurement units) in the future, enabled by advanced control, communication, and computation technologies [27], which means more and more information need to be estimated fast and accurately. As a result, the need for more decentralized estimation and control in smart grid systems will experience a high priority. State estimation is one of the key functions in control centers involving energy management

systems. A state estimator converts redundant meter readings and other available information obtained from a supervisory control and data acquisition system into an estimate of the state of an interconnected power system and distribution system [4].

In recent years, distributed signal processing has become a powerful tool to perform distributed state estimation in smart grids. To discuss the application in smart grids, we consider the IEEE 14–bus system [27], where 14 is the number of substations. At every time instant i, each bus $k, k = 1, 2, \ldots, 14$, takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = X_k\big(\boldsymbol{\omega}_0(i)\big) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{2.10}$$

where $\boldsymbol{\omega}_0(i)$ is the state vector of the entire interconnected system, $X_k(\boldsymbol{\omega}_0(i))$ is a nonlinear measurement function of bus k. The quantity $n_k(i)$ is the measurement error with mean equal to zero and which corresponds to bus k. Fig. 2.2 shows a standard IEEE–14 bus system with four non–overlapping control areas. Each control area will first perform the state estimation individually, and then exchange estimates through linked buses.

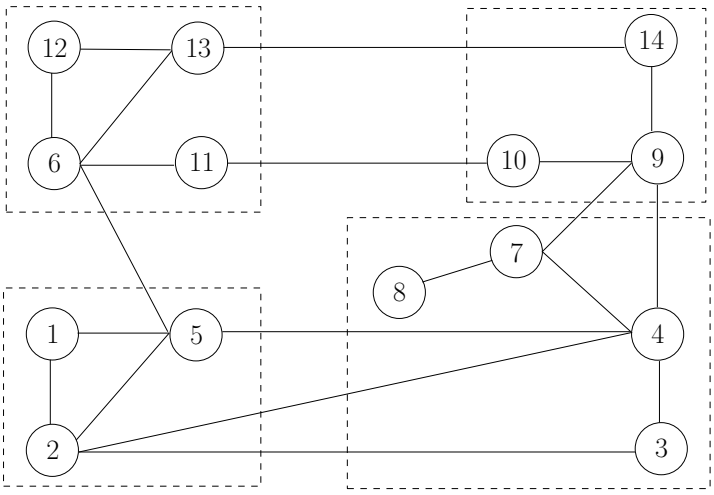

Figure 2.2: Example of the IEEE 14–bus system

Initially, we focus on the linearized Direct Current (DC) state estimation problem. Then, the state vector $\boldsymbol{\omega}_0(i)$ is measured as the voltage phase angle vector $\boldsymbol{\omega}_0$ for all buses. Therefore, the nonlinear measurement model for state estimation (2.10) is approximated by

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{2.11}$$

where $\boldsymbol{x}_k(i)$ is the measurement Jacobian vector for bus k which is known to the system. Then, the aim of the distributed estimation algorithm is to compute an estimate of $\boldsymbol{\omega}_0$ at each node, which can minimize the cost function given by

$$J_k(\boldsymbol{\omega}_k(i)) = \mathbb{E}|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)|^2. \tag{2.12}$$

Many distributed state estimation algorithms have been proposed in the literature to minimize the cost function (2.12), including $\mathcal{M}$–$\mathcal{CSE}$ algorithm [4], the single link strategy [28] and DESTA and DSITA algorithms [29]. In the later chapter, we compare the proposed algorithms with these existing strategies.

2.2 Protocols for Cooperation and Exchange of Information

As previously mentioned, there are three main cooperation strategies, incremental, diffusion and consensus [5] and recent studies indicate that the diffusion strategy is the most effective one [5]. In this section, we introduce and analyze each protocol for cooperation and exchange of information in detail.

2.2.1 Incremental Strategy

The incremental strategy is the simplest cooperation strategy. It works following a Hamiltonian cycle [1], the information goes through these nodes in one direction, which means each node passes the information to its adjacent node in a uniform direction. For the incremental strategy, starting from a given network topology, at each time instant i, every node k in the network will take a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{2.13}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ input signal vector, $n_k(i)$ is the noise sample at each node with zero mean and variance $\sigma_{n,k}^2$. Node k will use the scalar measurement $d_k(i)$ and the input signal vector $\boldsymbol{x}_k(i)$, together with the local estimate $\boldsymbol{\psi}_{k-1}(i)$ of unknown parameter $\boldsymbol{\omega}_0$

from its neighbor, to generate a local estimate $\boldsymbol{\psi}_k(i)$ of $\boldsymbol{\omega}_0$ through a distributed estimation strategy [1]. Then, the local estimate $\boldsymbol{\psi}_k(i)$ will be pushed to the next neighbor $k+1$ in one direction. The final estimate of the network will be equal to the final node's estimate. To illustrate the incremental processing, based on the traditional LMS algorithm, the incremental LMS algorithm updates the estimate at node k as [1]

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\psi}_{k-1}(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)]^*. \tag{2.14}$$

The resulting incremental implementation is briefly illustrated in Fig. 2.3.

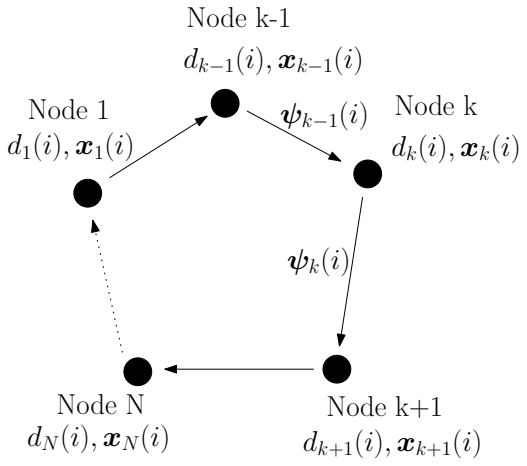

Figure 2.3: Incremental Distributed Estimation

2.2.2 Diffusion Strategy

For the diffusion strategy, the situation is different. Instead of getting information from one neighbor node, each node in the diffusion network will have some linked neighbors. There are two kinds of basic distributed diffusion estimation strategies, the Adapt–then–Combine (ATC) strategy and the Combine–then–Adapt (CTA) Strategy [17].

In the ATC strategy, each node will use adaptive algorithms such as LMS, RLS and CG to obtain a local estimate $\boldsymbol{\psi}_k(i)$. Then, each node will collect this estimate through all its neighbor nodes and combine them together to generate an updated estimate of $\boldsymbol{\omega}_0$ through

$$\boldsymbol{\omega}_k(i+1) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\psi}_l(i), \tag{2.15}$$

where c_{kl} are the combination coefficients calculated through the Metropolis, the Laplacian or the nearest neighbor rules [2], l indicates the neighbour node linked to node k and $\mathcal{N}_k$ denotes the set of neighbors of node k. The Metropolis rule is implemented as [30]

$$c_{kl} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}}, \text{ if } k \neq l \text{ are linked} \\ 1 - \sum\limits_{l \in \mathcal{N}_k \backslash k} c_{kl}, \text{ for } k = l, \end{cases} \tag{2.16}$$

where $|\mathcal{N}_k|$ denotes the cardinality of $\mathcal{N}_k$. The Laplacian rule is given by [31]

$$\boldsymbol{C} = \boldsymbol{I}_N - \kappa[\boldsymbol{D} - \boldsymbol{A}_d], \tag{2.17}$$

where $\boldsymbol{C}$ is the $N \times N$ combining coefficient matrix with entries $[c_{kl}]$, $\boldsymbol{D} = \text{diag}\{|\mathcal{N}_1|, |\mathcal{N}_2|, \ldots, |\mathcal{N}_N|\}$, $\kappa = 1/|\mathcal{N}_{max}|$, $|\mathcal{N}_{max}| = \max\{|\mathcal{N}_1|, |\mathcal{N}_2|, \ldots, |\mathcal{N}_N|\}$ and $\boldsymbol{A}_d$ is the $N \times N$ network adjacent matrix formed as

$$[A_d]_{kl} = \begin{cases} 1, \text{ if } k \text{ and } l \text{ are linked and } k = l \\ 0, \text{ otherwise.} \end{cases} \tag{2.18}$$

For the nearest neighbor rule, the combining coefficient c_{kl} is defined as [32]

$$c_{kl} = \begin{cases} \frac{1}{|\mathcal{N}_k|}, \text{ if } k \text{ and } l \text{ are linked} \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.19}$$

The combining coefficients c_{kl} should satisfy

$$\sum_{l \in \mathcal{N}_k \forall k} c_{kl} = 1. \tag{2.20}$$

Fig. 2.4 (a) describes the ATC diffusion strategy. Based on the traditional LMS algorithm, the diffusion LMS ATC strategy is illustrated in Table 2.1.

Table 2.1: Diffusion LMS ATC strategy

Initialize: $\boldsymbol{\omega}_k(1) = 0, k = 1, 2, \ldots, N$
for each time instant i
each node k performs the update:
$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*$.
then
$\boldsymbol{\omega}_k(i+1) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\psi}_l(i)$.
end

The CTA strategy just operates in a reverse way. Each node starts with collecting the estimates of their neighbors in the previous time instant and combines them together through

$$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\omega}_l(i). \tag{2.21}$$

After the $\boldsymbol{\psi}_k(i)$ is generated, each node k will employ the $\boldsymbol{\psi}_k(i)$ together with its $d_k(i)$ and $\boldsymbol{x}_k(i)$ to generate $\boldsymbol{\omega}_k(i+1)$. Based on the traditional LMS algorithm, the diffusion LMS CTA strategy is illustrated in Table 2.2.

Table 2.2: Diffusion LMS CTA strategy

Initialize: $\boldsymbol{\omega}_k(1) = 0, k = 1, 2, \ldots, N$
for each time instant i
each node k performs the update:
$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\omega}_l(i)$.
$\boldsymbol{\omega}_k(i+1) = \boldsymbol{\psi}_k(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_k^H(i)\boldsymbol{x}_k(i)]^*$.
end

Fig. 2.4 (b) illustrates the CTA process and the comparison of the main steps for these two strategies are summarised in Table 2.3.

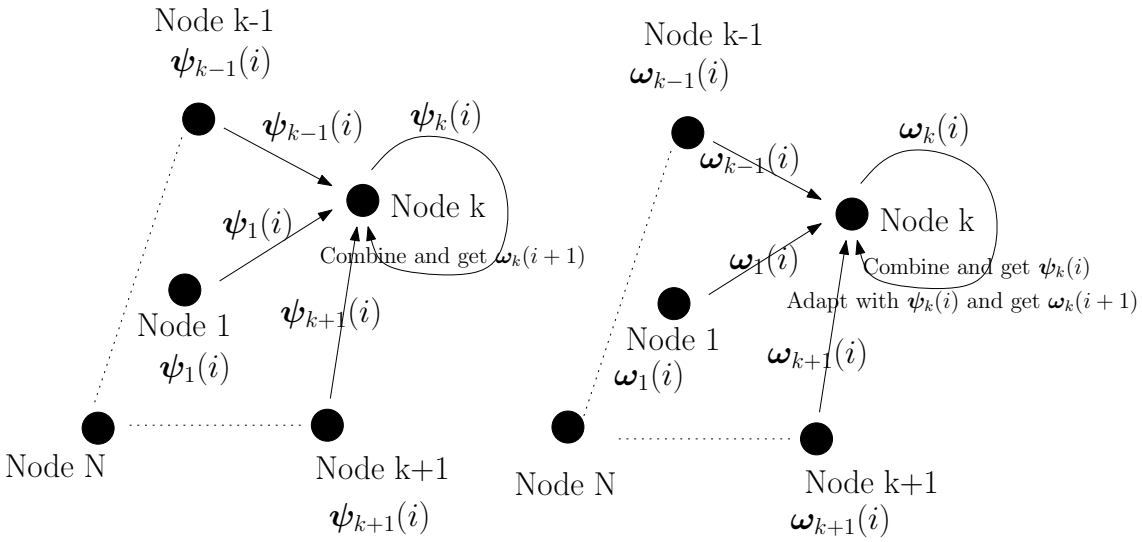

Figure 2.4: Diffusion Distributed Estimation

Table 2.3: Comparison of Main Steps for ATC and CTA strategies

Step	ATC Strategy	CTA Strategy
1	Adapt	Exchange $\boldsymbol{\omega}_l(i)$
2	Exchange $\boldsymbol{\psi}_l(i)$	Combine
3	Combine	Adapt

In general, for both CTA and ATC schemes, the combining coefficients c_{kl} determine which node $l \in \mathcal{N}_k$ should share their local estimates with node k, and this is a general convex combination. In [17], the simulation results show that the ATC scheme outperforms the CTA scheme.

2.2.3 Consensus Strategy

In the consensus strategy, each node will first collect all the previous estimates from all its neighbors and combine them together through the Metropolis, Laplacian or nearest neighbor rules to generate $\boldsymbol{\psi}_k(i)$. Then, each node will update its local estimate $\boldsymbol{\omega}_k(i+1)$ through adaptive algorithms (LMS, RLS, etc) with the combined estimate $\boldsymbol{\psi}_k(i)$ and its local estimate $\boldsymbol{\omega}_k(i)$.

Based on the traditional LMS algorithm, the LMS consensus strategy [14] is illustrated as follows:

for each time instant i

each node k performs the update:

$$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\omega}_l(i). \tag{2.22}$$

$$\boldsymbol{\omega}_k(i+1) = \boldsymbol{\psi}_k(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*. \tag{2.23}$$

end

The resulting consensus implementation is briefly illustrated in Fig. 2.5. If we compare the diffusion LMS CTA in Table 2.2 and LMS consensus (2.23), the only difference is the diffusion LMS CTA updates the estimate $\boldsymbol{\omega}_k(i+1)$ only through the combined estimate $\boldsymbol{\psi}_k(i)$, while the LMS consensus strategy employs $\boldsymbol{\psi}_k(i)$ and $\boldsymbol{\omega}_k(i)$ together, to update the estimate $\boldsymbol{\omega}_k(i+1)$.

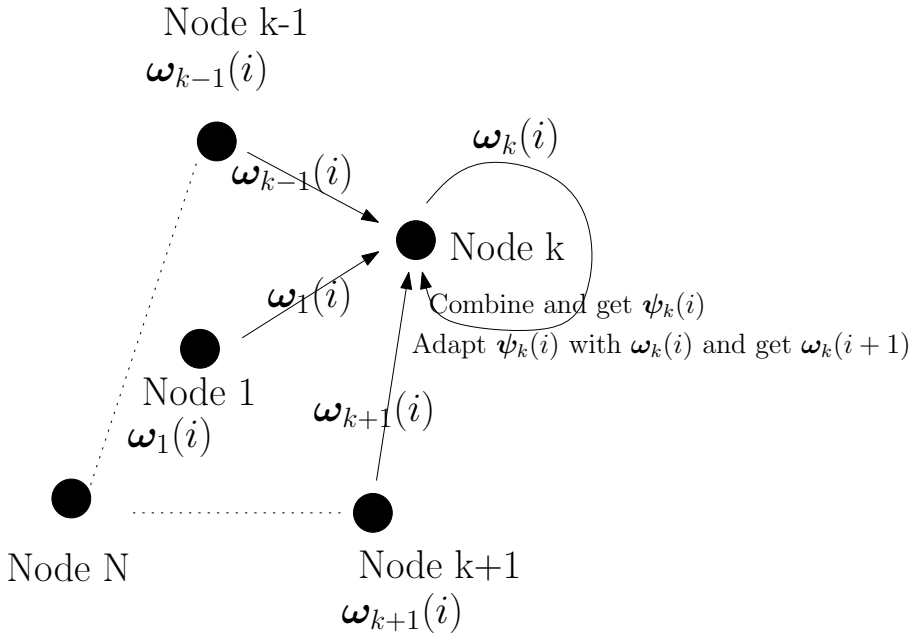

Figure 2.5: Consensus Distributed Estimation

2.3 Adaptive Algorithms

Adaptive signal processing is mainly based on algorithms that have the ability to learn by observing the environment and are guided by reference signals. An adaptive filter is a kind of digital filter which is able to automatically adjust the performance according to the input signal for various tasks such as estimation, prediction, smoothing and filtering. In most applications, designers employ finite impulse response (FIR) filters due to their inherent stability. In contrast, a non–adaptive filter has static coefficients and these static coefficients form the transfer function. According to the changes of the environment, the adaptive filter will use an adaptive algorithm to adjust the parameters. The most important characteristic of the adaptive filter is that it can work effectively in an unknown

environment and be able to track the time-varying features of the input signal [9]. With the performance enhancements of digital signal processors, adaptive filtering applications are becoming more common. Nowadays, they have been widely used in mobile phones and other communication devices, digital video recorders and digital cameras, as well as medical monitoring equipment [9, 10]. The basic structure of an adaptive filter is introduced in Fig. 2.6, where $\boldsymbol{x}(i)$ is the input signal vector, $y(i)$ and $d(i)$ are the output signal and reference signal, respectively, and $e(i)$ is the error signal which is calculated by $d(i) - y(i)$. Base on the basic ideas of adaptive signal processing, we develop several kinds of distributed signal processing strategies, which are introduced in the following chapters.

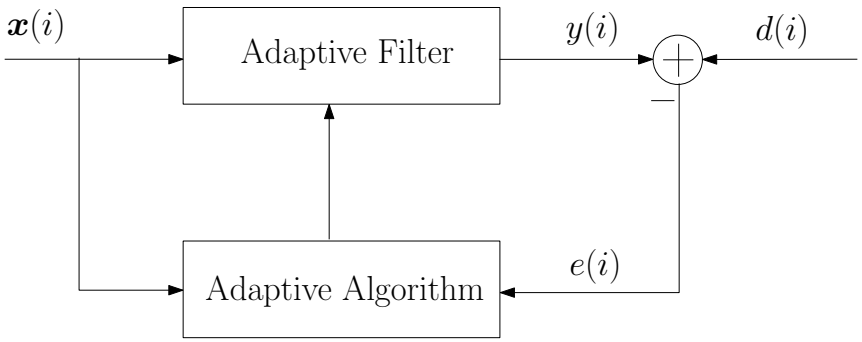

Figure 2.6: Adaptive Filter Structure

There are two widely used adaptive algorithms for adaptive signal processing, which are the least mean square (LMS) algorithm and the recursive least squares (RLS) algorithm [10]. The LMS algorithm is the simplest and the most basic algorithm for adaptive filters. It has the lowest computational complexity, however, its performance is not always satisfactory. The RLS has a better performance, but requires a high complexity. Additionally, the conjugate gradient (CG) algorithm is also a well known strategy for adaptive signal processing, as the CG algorithm has a faster convergence rate than the LMS–type algorithms and a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG employs [6–8]. In this section, the LMS, RLS and the CG algorithms are introduced.

2.3.1 The Least Mean Square (LMS) Algorithm

The LMS algorithm can be developed from the MSE cost function [9, 33]:

$$J(i) = \mathbb{E}|d(i) - \boldsymbol{\omega}^H(i)\boldsymbol{x}(i)|^2, \tag{2.24}$$

where $\mathbb{E}$ denotes expectation, $d(i)$ is the desired signal, $\boldsymbol{x}(i)$ is the input signal and $\boldsymbol{\omega}(i)$ is the tap weight vector. Then, the gradient vector of the cost function can be described as

$$\frac{\partial J\,(i)}{\partial \boldsymbol{\omega}^*\,(i)} = \boldsymbol{R}_x\boldsymbol{\omega}\,(i) - \boldsymbol{b}_x, \tag{2.25}$$

where $\boldsymbol{R}_x = \mathbb{E}|\boldsymbol{x}(i)\boldsymbol{x}^H(i)|$ is the input signal's correlation matrix and $\boldsymbol{b}_x = \mathbb{E}|\boldsymbol{x}(i)d^*(i)|$ stands for the crosscorrelation vector between the desired signal and the input signal. The optimum solution for the cost function (2.24) is the Wiener solution which is given by

$$\boldsymbol{\omega}_0\,(i) = \boldsymbol{R}_x^{-1}\boldsymbol{b}_x. \tag{2.26}$$

Since $\boldsymbol{R}_x$ and $\boldsymbol{b}_x$ are statistics of the received signal and are not given for the adaptive algorithms, these quantities must be estimated. LMS algorithms adopt the simplest estimator that uses instantaneous estimates for $\boldsymbol{R}_x$ and $\boldsymbol{b}_x$ [9, 33], which can be expressed mathematically as

$$\boldsymbol{R}_x = \boldsymbol{x}(i)\boldsymbol{x}^H(i), \tag{2.27}$$

$$\boldsymbol{b}_x = d^*(i)\boldsymbol{x}(i). \tag{2.28}$$

By plugging (2.27) and (2.28) into (2.25), we obtain

$$\frac{\partial J\,(i)}{\partial \boldsymbol{\omega}^*\,(i)} = -d^*(i)\boldsymbol{x}(i) + \boldsymbol{x}(i)\boldsymbol{x}^H(i)\boldsymbol{\omega}(i). \tag{2.29}$$

As a result, the corresponding filter coefficient vector is updated by

$$\begin{aligned}\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) - \mu\frac{\partial J(i)}{\partial \boldsymbol{\omega}^*(i)} \\ &= \boldsymbol{\omega}(i) + \mu\boldsymbol{x}(i)[d^*(i) - \boldsymbol{x}^H(i)\boldsymbol{\omega}(i)],\end{aligned} \tag{2.30}$$

where μ is the step size to control the speed of the convergence and steady–state performance.

2.3.2 The Recursive Least Squares (RLS) Algorithm

For the RLS algorithm, the cost function of the least squares error is shown as follows [9, 33]

$$\begin{aligned} J(i) &= \sum_{n=0}^{i} \lambda^{i-n} \epsilon\left(n\right)^2 \\ &= \sum_{n=0}^{i} \lambda^{i-n} |d(n) - \boldsymbol{\omega}^H(i)\boldsymbol{x}(n)|^2, \end{aligned} \tag{2.31}$$

where λ is the forgetting factor and $\epsilon(n)$ is the posteriori error at time instant n. After taking the gradient of $J(i)$ and letting it equal to zero, we obtain

$$\frac{\partial J\left(i\right)}{\partial \boldsymbol{\omega}^*\left(i\right)} = -2\sum_{n=0}^{i} \lambda^{i-n} \boldsymbol{x}\left(n\right)\left[d^*\left(n\right) - \boldsymbol{x}^H\left(n\right)\boldsymbol{\omega}\left(i\right)\right] = \mathbf{0}, \tag{2.32}$$

and

$$\boldsymbol{\omega}(i) = \left[\sum_{n=0}^{i} \lambda^{i-n} \boldsymbol{x}\left(n\right)^H \boldsymbol{x}\left(n\right)\right]^{-1} \sum_{n=0}^{i} \lambda^{i-n} \boldsymbol{x}\left(n\right) d^*\left(n\right). \tag{2.33}$$

To simplify (2.33), we define the following quantities:

$$\boldsymbol{\Phi}(i) = \sum_{n=0}^{i} \lambda^{i-n} \boldsymbol{x}\left(n\right) \boldsymbol{x}^H\left(n\right) \tag{2.34}$$

$$\boldsymbol{\theta}(i) = \sum_{n=0}^{i} \lambda^{i-n} \boldsymbol{x}\left(n\right) d^*\left(n\right). \tag{2.35}$$

and (2.33) becomes

$$\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i)^{-1}\boldsymbol{\theta}(i). \tag{2.36}$$

Then, (2.34) can be rewritten as

$$\begin{aligned} \boldsymbol{\Phi}(i) &= \lambda\left[\sum_{n=0}^{i-1} \lambda^{i-1-n} \boldsymbol{x}\left(n\right)^H \boldsymbol{x}\left(n\right)\right] + \boldsymbol{x}(i)\boldsymbol{x}^H(i) \\ &= \lambda\boldsymbol{\Phi}(i-1) + \boldsymbol{x}(i)\boldsymbol{x}^H(i). \end{aligned} \tag{2.37}$$

For (2.35), the same derivation process can be done and we get

$$\boldsymbol{\theta}(i) = \lambda\boldsymbol{\theta}(i-1) + \boldsymbol{x}(i)d^*(i). \tag{2.38}$$

Then, by using the matrix inversion lemma [9], we have

$$\boldsymbol{\Phi}^{-1}(i) = \lambda^{-1}\boldsymbol{\Phi}^{-1}(i-1) - \frac{\lambda^{-2}\boldsymbol{\Phi}^{-1}(i-1)\boldsymbol{x}(i)\boldsymbol{x}^H(i)\boldsymbol{\Phi}^{-1}(i-1)}{1 + \lambda^{-1}\boldsymbol{x}^H(i)\boldsymbol{\Phi}^{-1}(i-1)\boldsymbol{x}(i)}. \tag{2.39}$$

Let

$$\boldsymbol{P}(i) = \boldsymbol{\Phi}^{-1}(i) \tag{2.40}$$

and

$$\boldsymbol{k}(i) = \frac{\lambda^{-1}\boldsymbol{P}(i-1)\boldsymbol{x}(i)}{1+\lambda^{-1}\boldsymbol{x}^H(i)\boldsymbol{P}(i-1)\boldsymbol{x}(i)}. \tag{2.41}$$

Using (2.40) and (2.41), (2.36) and (2.39) can be respectively turned into

$$\boldsymbol{\omega}(i) = \boldsymbol{P}(i)\boldsymbol{\theta}(i) \tag{2.42}$$

and

$$\boldsymbol{P}(i) = \lambda^{-1}\boldsymbol{P}(i-1) - \lambda^{-1}\boldsymbol{k}(i)\boldsymbol{x}^H(i)\boldsymbol{P}(i-1). \tag{2.43}$$

In fact, (2.41) can be further rewritten as

$$\begin{aligned}\boldsymbol{k}(i) &= [\lambda^{-1}\boldsymbol{P}(i-1) - \lambda^{-1}\boldsymbol{k}(i)\boldsymbol{x}^H(i)\boldsymbol{P}(i-1)]\boldsymbol{x}(i) \\ &= \boldsymbol{P}(i)\boldsymbol{x}(i).\end{aligned} \tag{2.44}$$

Finally, when combining (2.38), (2.42) and (2.43) together

$$\begin{aligned}\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) - \boldsymbol{k}(i)\boldsymbol{x}^H(i)\boldsymbol{\omega}(i) + \boldsymbol{P}(i)\boldsymbol{x}(i)d^*(i) \\ &= \boldsymbol{\omega}(i) + \boldsymbol{k}(i)[d(i) - \boldsymbol{\omega}^H(i)\boldsymbol{x}(i)]^*.\end{aligned} \tag{2.45}$$

The derivation of the RLS algorithm is now complete.

2.3.3 The Conjugate Gradient (CG) Algorithm

The conjugate gradient (CG) algorithm is well known for its faster convergence rate than the LMS algorithm and a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG employs [6–8]. In adaptive filtering techniques, the CG algorithm applied to the system $\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{b}$, starts with an initial guess of the solution $\boldsymbol{\omega}(0)$, with an initial residual $\boldsymbol{g}(0) = \boldsymbol{b}$, and with an initial search direction that is equal to the initial residual: $\boldsymbol{p}(0) = \boldsymbol{g}(0)$, where $\boldsymbol{R}$ is the correlation matrix of the input signal and $\boldsymbol{b}$ is the cross–correlation vector between the desired signal and the input signal.

The strategy for the conjugate gradient method is that at step j, the residual $\boldsymbol{g}(j) = \boldsymbol{b} - \boldsymbol{R}\boldsymbol{\omega}(j)$ is orthogonal to the Krylov subspace generated by $\boldsymbol{b}$ [6], and therefore each residual is orthogonal to all the previous residuals. The residual is computed at each step.

The solution at the next step is achieved using a search direction that is a linear combination of the previous search directions, which for $\boldsymbol{\omega}(1)$ is just a combination between the previous and the current residual.

Then, the solution at step j, $\boldsymbol{\omega}(j)$, could be obtained through $\boldsymbol{\omega}(j-1)$ from the previous iteration plus a step size $\alpha(j)$ times the last search direction. The immediate benefit of the search directions is that there is no need to store all the previous search directions, only the search direction from the last step is needed. Using the orthogonality of the residuals to these previous search directions, the search is linearly independent of the previous directions. For the solution in the next step, a new search direction is computed, as well as a new residual and new step size. To provide an optimal approximate solution of $\boldsymbol{\omega}$, the step size $\alpha(j)$ is calculated as $\alpha(j) = \frac{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}{\boldsymbol{p}(j-1)^H \boldsymbol{R}\boldsymbol{p}(j-1)}$ according to [6–8]. The complete iterative formulas of the CG algorithm [6–8] are summarized in Table 2.4.

Table 2.4: Main Steps for CG algorithm

Initial conditions:
$\boldsymbol{\omega}(0) = 0$, $\boldsymbol{g}(0) = \boldsymbol{b}$, $\boldsymbol{p}(0) = \boldsymbol{g}(0)$
– Step size: $\alpha(j) = \frac{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}{\boldsymbol{p}(j-1)^H \boldsymbol{R}\boldsymbol{p}(j-1)}$
– Approximate solution: $\boldsymbol{\omega}(j) = \boldsymbol{\omega}(j-1) + \alpha(j)\boldsymbol{p}(j-1)$
– Residual: $\boldsymbol{g}(j) = \boldsymbol{g}(j-1) - \alpha(j)\boldsymbol{R}\boldsymbol{p}(j-1)$
– Improvement at step i: $\beta(j) = \frac{\boldsymbol{g}(j)^H \boldsymbol{g}(j)}{\boldsymbol{g}(j-1)^H \boldsymbol{g}(j-1)}$
– Search direction: $\boldsymbol{p}(j) = \boldsymbol{g}(j) + \beta(j)\boldsymbol{p}(j-1)$

2.4 Compressive Sensing Techniques

In recent years, reconstructing sparse signals from a small number of incoherent linear measurements has attracted a growing interest. In this context, a novel theory has been proposed in the literature as compressive sensing (CS) or compressed sampling [11, 12, 34]. CS techniques can provide a superior performance when compared with traditional signal reconstruction techniques under suitable conditions. The rationale behinds CS is that certain classes of sparse or compressible signals in some basis, where most of their coefficients are zero or small and only a few are large, can be exactly or sufficiently

accurately reconstructed with high probability [34]. The measurement process projects the signals onto a small set of vectors, which is incoherent with the sparsity basis. In subsection 2.4.2, we explain the incoherence condition in detail.

For the reconstruction, one of the original breakthroughs in CS [11, 35, 36] was to show that linear programming methods can be used to efficiently reconstruct the data signal with high accuracy. Since then, many alternative methods have been proposed as a faster or superior (in terms of reconstruction rate) alternative to these linear programming algorithms. One approach is to use matching pursuit techniques, which was originally proposed in [37]. Based on the original matching pursuit algorithm, a variety of algorithms have been proposed in the literature, such as orthogonal matching pursuit (OMP) [38], stagewise orthogonal matching pursuit (StOMP) [39], compressive sampling matching pursuit (CoSaMP) [40] and gradient pursuit algorithms [41].

2.4.1 Goal of Compressive Sensing

Consider a real valued, finite length, one dimensional, discrete time signal $\boldsymbol{x}$, which can be viewed as an $M \times 1$ column vector in $\mathbb{R}^M$ with elements $x[m], m = 1, 2, ..., M$. Any signal in $\mathbb{R}^M$ can be represented in terms of a basis of $M \times 1$ vectors $\{\boldsymbol{u}_i\}_{i=1}^M$. Let $\boldsymbol{U}$ be an $M \times M$ orthonormal matrix where the i–th column is the i–th basis vector $\boldsymbol{u}_i$. Then the signal $\boldsymbol{x} \in \mathbb{R}^M$ can be expressed as a linear combination of these basis vectors by

$$\boldsymbol{x} = \sum_{i=1}^{M} z_i \boldsymbol{u}_i \tag{2.46}$$

or

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{z}, \tag{2.47}$$

where $\boldsymbol{z}$ is the $M \times 1$ column vector of weighting coefficients $z_i = \langle \boldsymbol{x}, \boldsymbol{u}_i \rangle = \boldsymbol{u}_i^H \boldsymbol{x}$ and $\langle \cdot \rangle$ denotes inner product. It is clear that $\boldsymbol{x}$ and $\boldsymbol{z}$ are equivalent representations of the signal, with $\boldsymbol{x}$ in the time or space domain and $\boldsymbol{z}$ in the $\boldsymbol{U}$ domain.

The signal $\boldsymbol{x}$ is S–sparse if it is a linear combination of only S basis vectors. In other words, only S of the z_i coefficients in (2.46) or (2.47) are nonzero and $M - S$ are zero. When $S \ll M$, the signal $\boldsymbol{x}$ is compressible if the representation (2.46) or (2.47) has just a few large coefficients and many small or zero coefficients.

CS techniques deal with sparse signals by directly acquiring a compressed signal representation [11]. Under this situation, we consider a general linear measurement process that computes $D < M$ inner products between $\boldsymbol{x}$ and a collection of vectors $\{\gamma_j\}_{j=1}^{D}$ as in $y_j = \langle \boldsymbol{x}, \gamma_j \rangle$ [34]. Then, the measurements y_j are arranged in an $D \times 1$ vector $\boldsymbol{y}$ and the measurement vectors γ_j^H as rows in a $D \times M$ matrix $\boldsymbol{\Gamma}$. The matrix $\boldsymbol{\Gamma}$ is called measurement matrix. By substituting $\boldsymbol{U}$ from (2.47), $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = \boldsymbol{\Gamma x} = \boldsymbol{\Gamma U z} = \boldsymbol{\Theta z}, \tag{2.48}$$

where $\boldsymbol{\Theta} = \boldsymbol{\Gamma U}$ is an $D \times M$ matrix.

In conclusion, the goal of CS is that from D measurements where $D \ll M$ and $D \geq S$, the original signal $\boldsymbol{x}$ can be perfectly reconstructed, where the measurements are not chosen in an adaptive manner, meaning that the measurement matrix $\boldsymbol{\Gamma}$ is fixed and does not depend on the signal $\boldsymbol{x}$. To achieve this goal, the next step for compressive sensing is to design the measurement matrix $\boldsymbol{\Gamma}$ and the reconstruction algorithms.

2.4.2 Measurement Matrix

In this subsection, the design of a stable measurement matrix is discussed. The ultimate goal is to design the matrix $\boldsymbol{\Gamma}$ which does not destroy any information contained in the original signal $\boldsymbol{x}$. However since $\boldsymbol{\Gamma} \in \mathbb{R}^{D \times M}$ and $D < M$, it is not possible in general as Equation (2.48) is underdetermined, making the problem of solving for $\boldsymbol{x}$ or $\boldsymbol{z}$ ill–conditioned. If, however, $\boldsymbol{x}$ is S–sparse and the S locations of the nonzero coefficients in $\boldsymbol{z}$ are known, then the problem can be solved provided $D \geq S$. A necessary and sufficient condition for this simplified problem to be well conditioned is that, for any vector $\boldsymbol{v}$ sharing the same S nonzero entries as $\boldsymbol{z}$ and for some $\epsilon > 0$ [11, 34]

$$1 - \epsilon \leq \frac{||\boldsymbol{\Theta v}||_2}{||\boldsymbol{v}||_2} \leq 1 + \epsilon, \tag{2.49}$$

where $||\cdot||_2$ denotes the ℓ_2 norm. That is the matrix $\boldsymbol{\Theta}$ must preserve the lengths of these S–sparse vectors. However, it is unlikely that the positions of the non–zero elements are known in priori, but one can show that a sufficient condition for a stable solution for both S–sparse and compressible signals is that $\boldsymbol{\Theta}$ satisfies (2.49) for an arbitrary $3S$–sparse vector $\boldsymbol{v}$ [35]. This condition is referred to as the restricted isometry property (RIP). Apart

from the RIP condition, a related condition, referred to as incoherence, requires that the rows $\{\boldsymbol{\gamma}_j\}$ of $\boldsymbol{\Gamma}$ cannot sparsely represent the columns $\{\boldsymbol{u}_i\}$ of $\boldsymbol{U}$ [34], for example, $\boldsymbol{u}_i$ can not be a linear combination of $\boldsymbol{\gamma}_j$. In other words, the design of the measurement matrix requires a high degree of incoherence between the measurement matrix $\boldsymbol{\Gamma}$ and the basis matrix $\boldsymbol{U}$.

According to [11, 35, 42], both the RIP and incoherence conditions can be achieved with high probability simply by selecting $\boldsymbol{\Gamma}$ as a random matrix. The Gaussian measurement matrix $\boldsymbol{\Gamma}$ has two useful properties:

- The measurement matrix $\boldsymbol{\Gamma}$ is incoherent with the basis $\boldsymbol{U} = \boldsymbol{I}$ of delta spikes with high probability. More specifically, a $D \times M$ independent and identically distributed (iid) Gaussian matrix $\boldsymbol{\Theta} = \boldsymbol{\Gamma}\boldsymbol{I} = \boldsymbol{\Gamma}$ can be shown to have the RIP with high probability if $D \geq cS\log(M/S)$, with c a small constant [11, 35, 42]. Therefore, S–sparse and compressible signals of length M can be recovered from only $D \geq cS\log(M/S) \ll M$ random Gaussian measurements.
- The measurement matrix $\boldsymbol{\Gamma}$ is universal in the sense that $\boldsymbol{\Theta} = \boldsymbol{\Gamma}\boldsymbol{U}$ will be iid Gaussian matrix and thus have the RIP with high probability regardless of the choice of the orthonormal basis $\boldsymbol{U}$.

2.4.3 Reconstruction Algorithms

In this subsection, we present an overview of existing algorithms that can be used to reconstruct the signal $\boldsymbol{x}$ from the measured signal $\boldsymbol{y}$. In particular, we will focus on the Orthogonal Matching Pursuit (OMP) algorithm. In general, the signal reconstruction algorithm must take the D measurements in the vector $\boldsymbol{y}$, the random measurement matrix $\boldsymbol{\Gamma}$ and then reconstruct the length–M signal $\boldsymbol{x}$.

Matching pursuit is a class of iterative algorithms that decomposes a signal into a linear expansion of functions that form a dictionary. Matching pursuit was first introduced by Mallat and Zhang in [37]. At each iteration of the algorithm, matching pursuit chooses dictionary elements in a greedy fashion that best approximates the signal.

Orthogonal matching pursuit is an improved reconstruction algorithm based on matching pursuit. The principle behind OMP is similar to matching pursuit, at every iteration an element is chosen from the dictionary that best approximates the residual. However, instead of simply taking the scalar product of the residual and the new dictionary element, to calculate the coefficient weight, the original function will be fitted to all the already selected dictionary elements via least squares or projecting the function orthogonally onto all selected dictionary atoms [37, 43].

In the following, we describe the OMP algorithm in detail.
Input:

- A measurement matrix $\mathbf{\Gamma} \in \mathbb{R}^{D \times M}$.
- Observation vector $\boldsymbol{y} \in \mathbb{R}^{D}$.
- The sparsity level S of the target signal $\boldsymbol{x} \in \mathbb{R}^{M}$.
- According to (2.48), the signal model is $\boldsymbol{y} = \mathbf{\Gamma}\boldsymbol{x}$, where we set $\boldsymbol{U} = \boldsymbol{I}$.

Output:

- An estimate $\hat{\boldsymbol{x}} \in \mathbb{R}^{M}$ of the target signal $\boldsymbol{x}$.
- A set Λ_S containing the positions of the non–zero elements of $\hat{\boldsymbol{x}}$.
- An approximation $\boldsymbol{a}_S$ of the measurements $\boldsymbol{y}$.
- The residual $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{a}_S$.

Procedure:

1) Initialize the residual $\boldsymbol{r}_0 = \boldsymbol{y}$, the index set $\Lambda_S = \varnothing$, and the iteration counter $i = 1$. We define $\mathbf{\Gamma}_0$ as an empty matrix.

2) Find the index λ_i that solves the optimization problem

$$\lambda_i = \arg \max_{j=1,\ldots,M} |\langle \boldsymbol{r}_{i-1}, \boldsymbol{\gamma}_j \rangle|, \tag{2.50}$$

where $\boldsymbol{\gamma}_j$ is the column of $\boldsymbol{\Gamma}$. If the maximum occurs for multiple indices, each index path will be considered individually.

3) Augment the index set and the matrix of chosen atoms: $\Lambda_i = \Lambda_{i-1} \cup \{\lambda_i\}$ and $\boldsymbol{\Gamma}_i = [\boldsymbol{\Gamma}_{i-1} \ \ \boldsymbol{\gamma}_{\lambda_i}]$.

4) Solve a least squares problem to obtain a new signal estimate:

$$\boldsymbol{x}_i = \arg\min_{\hat{\boldsymbol{x}}} ||\boldsymbol{y} - \boldsymbol{\Gamma}_i \hat{\boldsymbol{x}}||_2. \tag{2.51}$$

5) Calculate the new approximation of the data and the new residual

$$\boldsymbol{a}_i = \boldsymbol{\Gamma}_i \boldsymbol{x}_i \tag{2.52}$$

$$\boldsymbol{r}_i = \boldsymbol{y} - \boldsymbol{a}_i. \tag{2.53}$$

6) Increment i, and return to Step 2 if $i < S$.

7) The estimate $\hat{\boldsymbol{x}}$ for the ideal signal has nonzero indices at the components listed in Λ_S. The value of the estimate $\hat{\boldsymbol{x}}$ in component λ_i equals the ith component of $\boldsymbol{x}_i$.

2.5 Sparsity–Aware Techniques

A sparsity–aware strategy is another technique that deals with sparse signals. In many situations, the signal of interest is sparse, containing only a few relatively large coefficients among many negligible ones. Any prior information about the sparsity of the signal of interest can be exploited to help improve the estimation performance, as demonstrated in many recent efforts in the area of CS [44]. However, the performance of most CS heavily relies on the recovery strategies, where the estimation of the desired vector is achieved from a collection of a fixed number of measurements.

Motivated by LASSO [45] and recent progress in compressive sensing [11, 12, 34], sparsity–aware strategies have been proposed in [3]. The basic idea of sparsity–aware strategies is to introduce a penalty which favors sparsity in the cost function. In this section, two kinds of sparsity–aware strategies are introduced, which are the Zero–Attracting strategy and the Reweighted Zero–Attracting strategy. First, an ℓ_1–norm penalty on the

coefficients is incorporated into the cost function. This results in a modified update step with a zero attractor for all the coefficients, which is the reason why this is called the Zero–Attracting strategy. Then, the Reweighted Zero–Attracting strategy has been proposed to further improve the performance. It employs reweighted step sizes of the zero attractor for different coefficients, inducing the attractor to selectively promote zero coefficients rather than uniformly promote zeros on all the coefficients. Details of these two strategies will be discussed in the following subsections.

2.5.1 The Zero–Attracting Strategy

In this subsection, the Zero–Attracting strategy is introduced in detail and derived based on the LMS algorithm. Starting from the cost function (2.24), a convex regularization term $f(\boldsymbol{\omega}(i))$ weighted by the parameter ρ is incorporated into the cost function, which results in:

$$J(i) = \mathbb{E}|d(i) - \boldsymbol{\omega}^H(i)\boldsymbol{x}(i)|^2 + \rho f(\boldsymbol{\omega}(i)), \tag{2.54}$$

where $f(\boldsymbol{\omega}(i))$ is used to enforce sparsity. Based on (2.30) and using the gradient descent updating, the corresponding filter coefficient vector is updated by

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \mu\boldsymbol{x}(i)[d(i) - \boldsymbol{\omega}^H(i)\boldsymbol{x}(i)]^* - \mu\rho\partial f(\boldsymbol{\omega}(i)). \tag{2.55}$$

This results in a modified LMS update with a zero attractor for all the coefficients, naming the Zero–Attracting LMS (ZA–LMS) algorithm. According to [3], for the ZA–LMS algorithm, the ℓ_1–norm is proposed as penalty function, i.e.,

$$f_1(\boldsymbol{\omega}(i)) = \|\boldsymbol{\omega}(i)\|_1, \tag{2.56}$$

in the new cost function (2.54). This choice leads to an algorithm update in (2.55) where the subgradient vector is given by $\partial f_1(\boldsymbol{\omega}(i)) = \text{sign}(\boldsymbol{\omega}(i))$, where $\text{sign}(a)$ is a component–wise function defined as

$$\text{sign}(a) = \begin{cases} a/|a| & a \neq 0 \\ 0 & a = 0. \end{cases} \tag{2.57}$$

The Zero–Attracting update uniformly shrinks all coefficients of the vector, and does not distinguish between zero and non–zero elements. Intuitively, the zero attractor will speed–up convergence when the majority of coefficients of $\boldsymbol{\omega}_0$ are zero, i.e., the system is

sparse. However, since all the coefficients are forced toward zero uniformly, the performance would deteriorate for systems that are not sufficiently sparse.

2.5.2 The Reweighted Zero–Attracting Strategy

Apart from the Zero–Attracting strategy, the Reweighted Zero–Attracting strategy is another kind of sparsity–aware technique. Motivated by the idea of reweighting in compressive sampling [46], the term $f(\boldsymbol{\omega}(i))$ is now defined as

$$f_2(\boldsymbol{\omega}(i)) = \sum_{m=1}^{M} \log(1 + \varepsilon|\omega_m(i)|). \tag{2.58}$$

The log–sum penalty $\sum_{m=1}^{M} \log(1 + \varepsilon|\omega_m(i)|)$ has been introduced as it behaves more similarly to the ℓ_0–norm than the ℓ_1–norm [46]. Thus, it enhances the sparsity recovery of the algorithm. The algorithm in (2.55) is then updated by using

$$\partial f_2(\boldsymbol{\omega}(i)) = \varepsilon \frac{\text{sign}(\boldsymbol{\omega}(i))}{1 + \varepsilon|\boldsymbol{\omega}(i)|}, \tag{2.59}$$

leading to what we shall refer to as the reweighted zero–attracting LMS (RZA–LMS) algorithm. The reweighted zero attractor of the RZA–LMS takes effect only on those coefficients whose magnitudes are comparable to $1/\varepsilon$ and there is little shrinkage exerted on the coefficients whose $|\omega_m(i)| \gg 1/\varepsilon$.

2.5.3 Simulation results

In this subsection, the performance of the ZA–LMS and the RZA–LMS algorithms are compared with the standard LMS algorithm in terms of their MSD performance, based on a system identification scenario.

System identification involves the following steps: experimental planning, the selection of a model structure, parameter estimation, and model validation [9]. Suppose we have an unknown plant $\boldsymbol{\omega}_0$ which is linear. $\boldsymbol{x}(i)$ is the available input signal at time instant i with size $M \times 1$ and is applied simultaneously to the plant and the model. Then, $d(i)$ and $y(i)$ are employed to stand for the desired signal and the output of the adaptive

filter respectively. The output $y(i)$ is obtained by

$$y(i) = \boldsymbol{\omega}^H(i)\boldsymbol{x}(i), \tag{2.60}$$

where $\boldsymbol{\omega}(i)$ is the estimate of the unknown plant generated by the adaptive filter. The error $e(i) = d(i) - y(i)$ is used to adjust the adaptive filter. The task of the adaptive filtering algorithm is to keep the modeling error small [33]. The structure for the system identification is shown in Fig. 2.7.

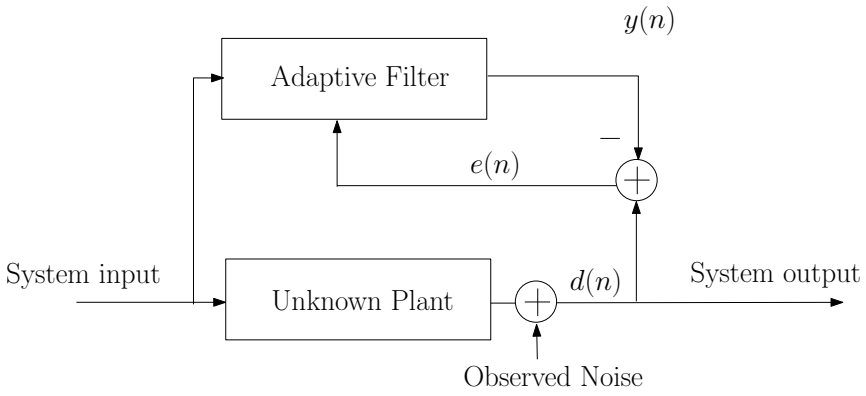

Figure 2.7: System identification

The unknown plant $\boldsymbol{\omega}_0$ contains 20 coefficients and three scenarios are considered. In the first scenario, we set the 5th coefficient with value 1 and others to zero, forcing the system to have a sparsity level of 1/20. In the second scenario, all the odd coefficients are set to 1 while all the even coefficients remaining to be zero, i.e., a sparsity level of 10/20. In the third scenario, all the even coefficients are set with value -1 while all the odd coefficients are maintained to be 1, leaving a completely non–sparse system. The input signal and the observed noise are white Gaussian random sequences with variance of 1 and 10^{-3}, respectively. The parameters are set as $\mu = 0.05$, $\rho = 5 \times 10^{-4}$ and $\varepsilon = 10$. Note that the same μ and ρ are used for the LMS, ZA–LMS and RZA–LMS algorithms. The average mean square deviation (MSD) is shown in Fig. 2.8, 2.9 and 2.10.

It is clear that, from Fig. 2.8, when the system is very sparse, both ZA–LMS and RZA–LMS yield faster convergence rate and better steady–state performances than the LMS algorithm. The RZA–LMS algorithm achieves lower MSD than ZA–LMS. In the second scenario, as the number of non–zero coefficients increases to 10, the performance of ZA–LMS deteriorates while RZA–LMS maintains the best performance among the

three algorithm. In the third scenario, RZA–LMS still performs comparably with the standard LMS algorithm even though the system is now completely non–sparse.

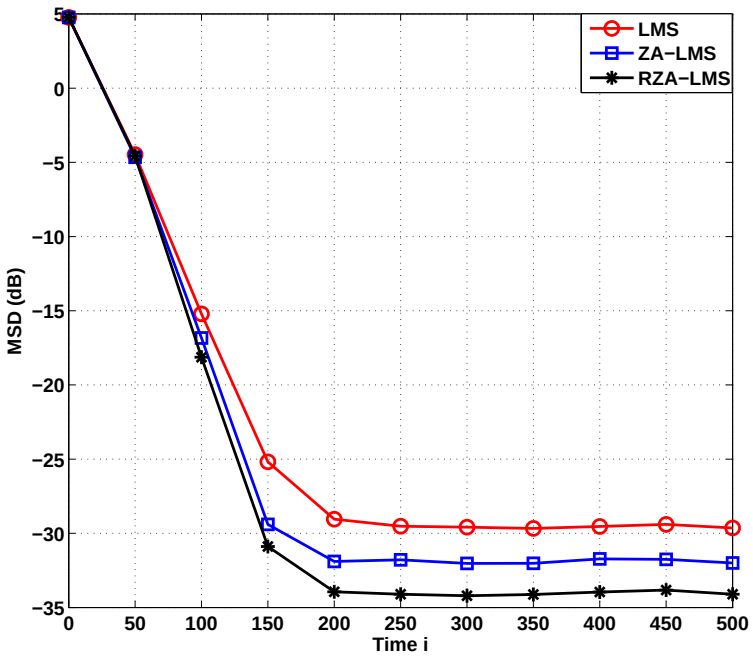

Figure 2.8: MSD comparison for scenario 1

2.6 Summary

In this chapter, we have presented an overview of the theory relevant to this thesis and introduced the applications that are used to present the work in this thesis, including distributed estimation, spectrum estimation and smart grids. The topics of distributed signal processing, incremental and diffusion strategies, adaptive signal processing, compressive sensing and sparsity–aware techniques are then covered and discussed with an outline of previous work in these fields and important applications.

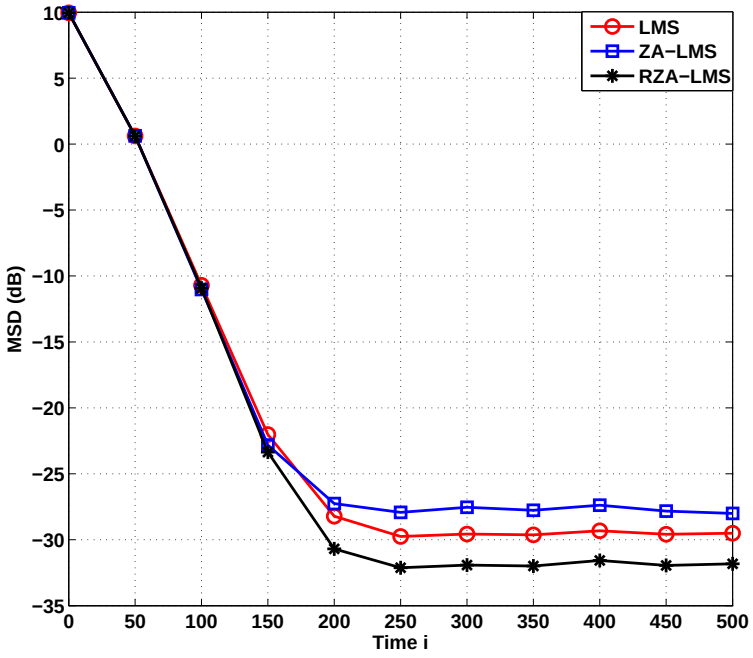

Figure 2.9: MSD comparison for scenario 2

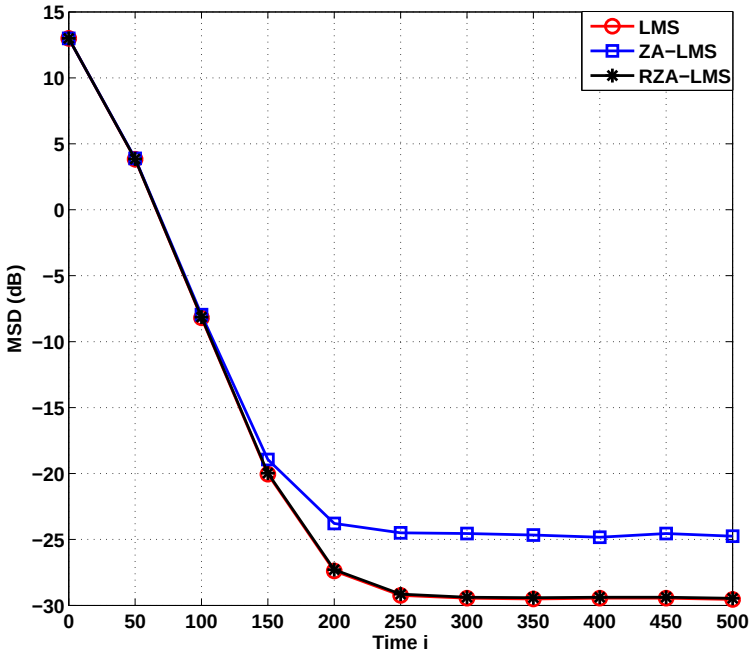

Figure 2.10: MSD comparison for scenario 3

Chapter 3

Distributed Conjugate Gradient Strategies for Distributed Estimation Over Sensor Networks

Contents

3.1 Introduction 36

3.2 System Models 38

3.3 Proposed Incremental Distributed CG–Based Algorithms 41

3.4 Proposed Diffusion Distributed CG–Based Algorithms 46

3.5 Preconditioner Design 48

3.6 Simulation Results 52

3.7 Summary 58

3.1 Introduction

In recent years, distributed processing has become popular in wireless networks. Distributed processing of information consists of collecting data at each node of a network of sensing devices spread over a geographical area, conveying information to the whole network and performing statistical inference in a distributed way [1, 47]. These techniques

exhibit several distinctive advantages such as flexibility, robustness to sensor failures and improved performance. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit their estimates to the specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved estimate. There are three main protocols for cooperation and exchange of information for distributed processing, incremental, diffusion and consensus strategies, and recent studies indicate that the diffusion strategy is the most effective one [5].

In the last few years, several algorithms have been developed and reported in the literature for distributed networks. Steepest-descent, least mean square (LMS) [1], recursive least squares (RLS) [48] and affine projection (AP) [16] solutions have been considered with incremental adaptive strategies over distributed networks [1], while LMS, AP and recursive least squares (RLS) algorithms have been reported using diffusion adaptive strategies [2, 18, 20, 49, 50]. Although the LMS–based algorithms have their own advantages, when compared with conjugate gradient (CG) algorithms, there are several disadvantages. First, for the LMS–based algorithms, the adaptation speed is often slow, especially for the conventional LMS algorithm. Second, with the increase of the adaptation speed, the system stability may decrease significantly [13]. Furthermore, the RLS–based algorithms usually are prone to numerical instability when implemented in hardware [9]. In order to develop distributed solutions with a more attractive tradeoff between performance and complexity, we focus on the development of distributed CG algorithms. To the best of our knowledge, CG–based algorithms have not been developed so far for distributed processing. The existing standard CG algorithm has a faster convergence rate than the LMS-type algorithms and a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG employs [6–8]. We consider variants of CG algorithms, including the conventional CG (CCG) and modified CG (MCG) algorithms [8, 51].

In this chapter, we propose distributed CG algorithms for both incremental and diffusion adaptive strategies. In particular, we develop distributed versions of the CCG algorithm and of the MCG algorithm for distributed estimation and spectrum estimation using wireless sensor networks. The design of preconditioners for CG algorithms, which have the ability to improve the performance of the CG algorithms is also presented in this chapter. These algorithms can be applied to civilian and defence applications, such

as parameter estimation in wireless sensor networks, biomedical engineering, cellular networks, battlefield information identification, movement estimation and detection and distributed spectrum estimation.

In summary, the main contributions of this chapter are:

- We devise distributed CCG and MCG algorithms for incremental and diffusion strategies to perform distributed estimation tasks.
- The design of preconditioners for CG algorithms, which have the ability to improve the performance of the proposed CG algorithms.
- A simulation study of the proposed and existing distributed estimation algorithms with applications to distributed parameter estimation and spectrum estimation.

3.2 System Models

In this section, we describe the system models of two applications of distributed signal processing, namely, parameter estimation and spectrum estimation. In these applications, we consider a wireless sensor network which employs distributed signal processing techniques to perform the desired tasks. We consider a set of N nodes, distributed over a given geographical area. The nodes are connected and form a network, which is assumed to be partially connected because nodes can exchange information only with neighbors determined by the connectivity topology. We call a network with this property a partially connected network whereas a fully connected network means that data broadcast by a node can be captured by all other nodes in the network in one hop [19].

3.2.1 Distributed Parameter Estimation

For distributed parameter estimation, we focus on the processing of estimating an unknown vector $\boldsymbol{\omega}_0$ with size $M \times 1$. The desired signal of each node at time instant i

is

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{3.1}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ input signal vector, $n_k(i)$ is the Gaussian noise at each node with zero mean and variance $\sigma_{n,k}^2$. At the same time, the output of the adaptive algorithm for each node is given by

$$y_k(i) = \boldsymbol{\omega}_k^H(i) \boldsymbol{x}_k(i), \quad i = 1, 2, \ldots, I, \tag{3.2}$$

where $\boldsymbol{\omega}_k(i)$ is the local estimate of $\boldsymbol{\omega}_0$ for each node at time instant i.

To compute an estimate of the unknown vector, we need to solve a problem expressed in the form of a minimization of the cost function in the distributed form for each node k:

$$J_k\big(\boldsymbol{\omega}_k(i)\big) = \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2 \tag{3.3}$$

and the global network cost function could be described as

$$J\big(\boldsymbol{\omega}_k(i)\big) = \sum_{k=1}^{N} \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2. \tag{3.4}$$

The optimum solution for the cost function (3.3) is the Wiener solution which is given by

$$\boldsymbol{\omega}_0 = \boldsymbol{R}_k^{-1}(i)\boldsymbol{b}_k(i). \tag{3.5}$$

where the $M \times M$ autocorrelation matrix is given by $\boldsymbol{R}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]$ and $\boldsymbol{b}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)d_k^*(i)]$ is an $M \times 1$ cross–correlation matrix. In this chapter, we focus on incremental and diffusion CG–based algorithms to solve the equation and perform parameter estimation and spectrum estimation in a distributed fashion.

3.2.2 Distributed Spectrum Estimation

In distributed spectrum estimation, we aim to estimate the spectrum of a transmitted signal $\boldsymbol{s}$ with N nodes using a wireless sensor network. Let $\boldsymbol{\Phi}_s(f)$ denote the power spectral density (PSD) of the signal $\boldsymbol{s}$. The PSD can be represented as a linear combination of some $\mathcal{B}$ basis functions, as described by

$$\Phi_s(f) = \sum_{m=1}^{\mathcal{B}} b_m(f)\omega_{0m} = \boldsymbol{b}_0^T(f)\boldsymbol{\omega}_0, \tag{3.6}$$

where $\boldsymbol{b}_0(f) = [b_1(f), ..., b_{\mathcal{B}}(f)]^T$ is the vector of basis functions evaluated at frequency f, $\boldsymbol{\omega}_0 = [\omega_{01}, ..., \omega_{0\mathcal{B}}]$ is a vector of weighting coefficients representing the power that transmits the signal $\boldsymbol{s}$ over each basis, and $\mathcal{B}$ is the number of basis functions. For $\mathcal{B}$ sufficiently large, the basis expansion in (3.6) can well approximate the transmitted spectrum. Possible choices for the set of basis $\{b_m(f)\}_{m=1}^{\mathcal{B}}$ include [22–24]: rectangular functions, raised cosines, Gaussian bells and Splines.

Let $H_k(f, i)$ be the channel transfer function between a transmit node conveying the signal $\boldsymbol{s}$ and receive node k at time instant i, the PSD of the received signal observed by node k can be expressed as

$$\begin{aligned} I_k(f,i) &= |H_k(f,i)|^2 \Phi_s(f) + v_k^2 \\ &= \sum_{m=1}^{\mathcal{B}} |H_k(f,i)|^2 b_m(f) \omega_{0m} + v_k^2 \\ &= \boldsymbol{b}_{k,i}^T(f) \boldsymbol{\omega}_0 + v_k^2 \end{aligned} \tag{3.7}$$

where $\boldsymbol{b}_{k,i}^T(f) = [|H_k(f,i)|^2 b_m(f)]_{m=1}^{\mathcal{B}}$ and v_k^2 is the receiver noise power at node k.

At every time instant i, every node k observes measurements of the noisy version of the true PSD $I_k(f, i)$ described by (3.7) over N_c frequency samples $f_j = f_{min} : (f_{max} - f_{min})/N_c : f_{max}$, for $j = 1, ..., N_c$, according to the model:

$$d_k^j(i) = \boldsymbol{b}_{k,i}^T(f_j) \boldsymbol{\omega}_0 + v_k^2 + n_k^j(i). \tag{3.8}$$

The term $n_k^j(i)$ denotes sampling noise and have zero mean and variance $\sigma_{n,j}^2$. The receiver noise power $v_{n,k}^2$ can be estimated with high accuracy in a preliminary step using, e.g., an energy estimator over an idle band, and then subtracted from (3.8) [25, 26]. Thus, collecting measurements over N_c contiguous channels, we obtain a linear model given by

$$\boldsymbol{d}_k(i) = \boldsymbol{B}_k(i) \boldsymbol{\omega}_0 + \boldsymbol{n}_k(i), \tag{3.9}$$

where $\boldsymbol{B}_k(i) = [\boldsymbol{b}_{k,i}^T(f_j)]_{j=1}^{N_c} \in \mathbb{R}^{N_c \times \mathcal{B}}$, with $N_c > \mathcal{B}$, and $\boldsymbol{n}_k(i) = [n_k^1(i), ...n_k^{N_c}(i)]^T$. At this point, we can generate the cost function for node k as:

$$J_{\boldsymbol{\omega}_k(i)}(\boldsymbol{\omega}_k(i)) = \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}_k(i)\big|^2 \tag{3.10}$$

and the global network cost function could be described as

$$J_{\boldsymbol{\omega}}\big(\boldsymbol{\omega}\big) = \sum_{k=1}^{N} \mathbb{E}\big|\boldsymbol{d}_k(i) - \boldsymbol{B}_k(i)\boldsymbol{\omega}\big|^2. \tag{3.11}$$

3.3 Proposed Incremental Distributed CG–Based Algorithms

In this section, we propose two CG–based algorithms which are based on the CCG [8] and MCG [52] algorithms with incremental distributed solution for distributed parameter estimation and spectrum estimation over wireless sensor networks.

3.3.1 Incremental Distributed CG–Based Solutions

In the incremental distributed strategy, each node is only allowed to communicate with its direct neighbor at each time instant. To describe the whole process, we define a cycle where each node in this network could only access its immediate neighbor in this cycle [1]. The quantity $\boldsymbol{\psi}_k(i)$ is defined as a local estimate of the unknown vector $\boldsymbol{\omega}_0$ at time instant i. As a result, we assume that node k has access to an estimate of $\boldsymbol{\omega}_0$ at its immediate neighbor node $k-1$ which is $\boldsymbol{\psi}_{k-1}(i)$ in the defined cycle. Fig.3.1 illustrates this processing. In the following, we introduce two kinds of incremental distributed CG–

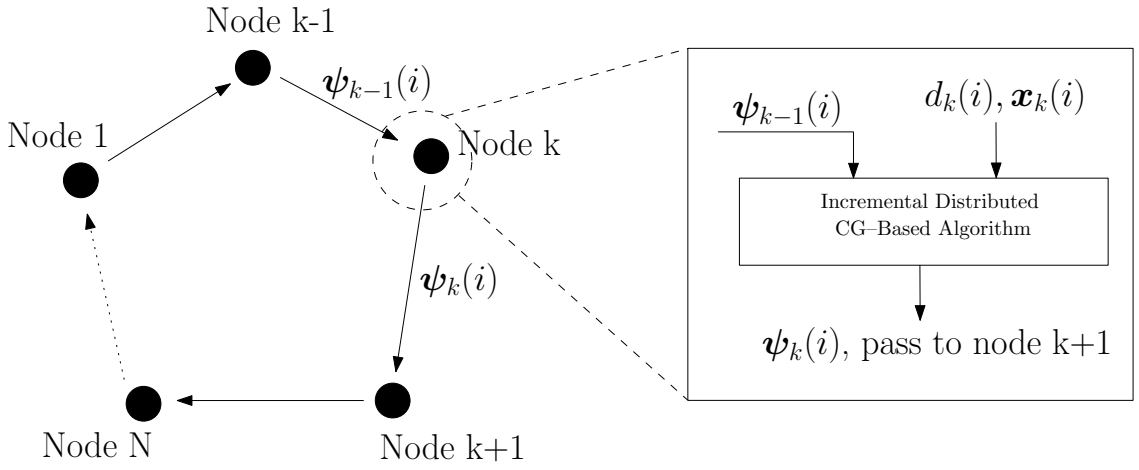

Figure 3.1: Incremental distributed CG–based network processing

based algorithms, which are the incremental distributed CCG (IDCCG) algorithm and the incremental distributed MCG (IDMCG) algorithm.

Proposed IDCCG Algorithm

Based on the main steps of CG algorithm which are described in Table. 2.4, we introduce the main steps of the proposed IDCCG algorithm. In the IDCCG algorithm, the iteration procedure is introduced. At the jth iteration of time instant i, the step size $\alpha_k^j(i)$ for updating the local estimate at node k is defined as:

$$\alpha_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H \boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)}, \tag{3.12}$$

where $\boldsymbol{p}_k^j(i)$ is the search direction and defined as

$$\boldsymbol{p}_k^j(i) = \boldsymbol{g}_k^j(i) + \beta_k^j(i)\boldsymbol{p}_k^{j-1}(i). \tag{3.13}$$

In (3.13), the coefficient $\beta_k^j(i)$ is calculated by the Gram–Schmidt orthogonalization procedure [7] for the conjugacy:

$$\beta_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j}(i)\right)^H \boldsymbol{g}_k^{j}(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}. \tag{3.14}$$

$\boldsymbol{g}_k^j(i)$ is the residual, which is obtained as

$$\boldsymbol{g}_k^j(i) = \boldsymbol{g}_k^{j-1}(i) - \alpha_k^j(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i). \tag{3.15}$$

The initial search direction is equal to the initial residual, which is given by $\boldsymbol{p}_k^0(i) = \boldsymbol{g}_k^0(i) = \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i)\boldsymbol{\psi}_k^0(i)$. Then, the local estimate is updated as

$$\boldsymbol{\psi}_k^j(i) = \boldsymbol{\psi}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i). \tag{3.16}$$

There are two ways to compute the correlation and cross–correlation matrices which are the 'finite sliding data window' and the 'exponentially decaying data window' [8]. In this chapter, we focus on the 'exponentially decaying data window'. The recursions are given by:

$$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i) \tag{3.17}$$

and

$$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i) \tag{3.18}$$

where λ_f is the forgetting factor. The IDCCD algorithm is summarized in Table 3.1

Table 3.1: IDCCG Algorithm

Initialization:	Complexity	Complexity
$\boldsymbol{\omega}(0) = \mathbf{0}$	Multiplications	Additions
For each time instant $i = 1, 2, \ldots, I$		
$\boldsymbol{\psi}_1^0(i) = \boldsymbol{\omega}(i-1)$		
For each node $k = 1, 2, \ldots, N$		
$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$	$2M^2$	M^2
$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i)$	$2M$	M
$\boldsymbol{p}_k^0(i) = \boldsymbol{g}_k^0(i) = \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i)\boldsymbol{\psi}_k^0(i)$	M^2	$2M^2$
For iterations $j = 1, 2, \ldots, J$		
$\alpha_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H \boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)}$	$M^2 + 2M + 1$	$M^2 + M - 2$
$\boldsymbol{\psi}_k^j(i) = \boldsymbol{\psi}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
$\boldsymbol{g}_k^j(i) = \boldsymbol{g}_k^{j-1}(i) - \alpha_k^j(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
$\beta_k^j(i) = \frac{\left(\boldsymbol{g}_k^j(i)\right)^H \boldsymbol{g}_k^j(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}$	$M + 1$	$M - 1$
$\boldsymbol{p}_k^j(i) = \boldsymbol{g}_k^j(i) + \beta_k^j(i)\boldsymbol{p}_k^{j-1}(i)$	M	M
End		
When $k < N$		
$\boldsymbol{\psi}_{k+1}^0(i) = \boldsymbol{\psi}_k^J(i)$		
End		
$\boldsymbol{\omega}(i) = \boldsymbol{\psi}_N^J(i)$		
End		

Proposed IDMCG Algorithm

The idea of the IDMCG algorithm comes from the existing MCG algorithm. For the IDMCG solution, a recursive formulation for the residual vector is employed, which can be found by using (3.12), (3.17) and (3.18) [8, 51], resulting in

$$\begin{aligned}\boldsymbol{g}_k(i) &= \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i)\boldsymbol{\psi}_k(i) \\ &= \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1) + \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)].\end{aligned} \tag{3.19}$$

Premultiplying (3.19) by $\boldsymbol{p}_k^H(i-1)$ gives

$$\begin{aligned}\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i) = \lambda_f \boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1) \\ + \boldsymbol{p}_k^H(i-1)\boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)].\end{aligned} \tag{3.20}$$

Taking the expectation of both sides and considering $\boldsymbol{p}_k(i-1)$ uncorrelated with $\boldsymbol{x}_k(i)$, $d_k(i)$ and $\boldsymbol{\psi}_{k-1}(i)$ yields

$$\begin{aligned}\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i)] \approx \lambda_f \mathbb{E}[\boldsymbol{p}_k(i-1)^H \boldsymbol{g}_{k-1}(i)] - \mathbb{E}[\alpha_k(i)]\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)] \\ + \mathbb{E}[\boldsymbol{p}_k^H(i-1)]\mathbb{E}\big[\boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\omega}_{k-1}^H(i)\boldsymbol{x}_k(i)]\big].\end{aligned} \tag{3.21}$$

Assuming that the algorithm converges, the last term of (3.21) can be neglected and we obtain [8]:

$$\mathbb{E}[\alpha_k(i)] = \frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i)] - \lambda_f \mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]} \tag{3.22}$$

and the $\mathbb{E}[\alpha_k(i)]$ should satisfied the following equation [8]:

$$(\lambda_f - 0.5)\frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]} \leq \mathbb{E}[\alpha_k(i)] \leq \frac{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)]}{\mathbb{E}[\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)]} \tag{3.23}$$

The inequalities in (3.23) are satisfied if we define [8]:

$$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}, \tag{3.24}$$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$. The direction vector $\boldsymbol{p}_k(i)$ for the IDMCG algorithm is defined by

$$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1). \tag{3.25}$$

Table 3.2: IDMCG Algorithm

Initialization:	Complexity	Complexity
$\boldsymbol{\omega}(0) = \mathbf{0}$	Multiplications	Additions
For each node $k = 1, 2, \ldots, N$		
$\boldsymbol{b}_k(1) = d_k^*(1)\boldsymbol{x}_k(1)$		
$\boldsymbol{p}_k(0) = \boldsymbol{g}_k(0) = \boldsymbol{b}_k(1)$		
End		
For each time instant $i = 1, 2, \ldots, I$		
$\boldsymbol{\psi}_0(i) = \boldsymbol{\omega}(i-1)$		
For each node $k = 1, 2, \ldots, N$		
$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$	$2M^2$	M^2
$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}$	$M^2 + 2M + 1$	$M^2 + M - 2$
where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$		
$\boldsymbol{\psi}_k(i) = \boldsymbol{\psi}_{k-1}(i) + \alpha_k(i)\boldsymbol{p}_k(i-1)$	M	M
$\boldsymbol{g}_k(i) = \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)$		
$+\boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)]$	$4M$	$3M$
$\beta_k(i) = \frac{\left(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\right)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)}$	$2M + 1$	$3M - 2$
$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1)$	M	M
End		
$\boldsymbol{\omega}(i) = \boldsymbol{\psi}_N(i)$		
End		

For the IDMCG algorithm, for the computation of $\beta_k(i)$, the Polak–Ribiere method [8], which is given by

$$\beta_k(i) = \frac{\left(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\right)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)} \tag{3.26}$$

should be used for improved performance, according to [53, 54].

In the comparison of the IDCCG algorithm with the IDMCG algorithm, the difference between these two strategies is that IDCCG needs to run J iterations while IDMCG only needs one iteration. The details of the IDMCG solution are shown in Table 3.2.

Table 3.3: Computational Complexity of Different Incremental Algorithms per Node

Algorithm	Additions	Multiplications
IDCCG	$2M^2 + M$ $+J(M^2 + 5M - 4)$	$3M^2 + 2M$ $J(M^2 + 6M + 2)$
IDMCG	$2M^2 + 9M - 4$	$3M^2 + 10M + 2$
Incremental LMS [1]	$4M - 1$	$3M + 1$
Incremental RLS [1]	$4M^2 + 12M + 1$	$4M^2 + 12M - 1$

3.3.2 Computational Complexity

To analyze the proposed incremental distributed CG algorithms, we detail the computational complexity in terms of arithmetic operations. Additions and multiplications are used to measure the complexity and are listed in Table 3.3. The parameter M is the length of the unknown vector $\boldsymbol{\omega}_0$ that needs to be estimated. Detailed computational complexity of each step for the proposed incremental distributed CG algorithms are shown in Table 3.1 and 3.2. When compare the IDMCG and IDCCG algorithms with the RLS–type algorithm, we can see that the complexity of IDMCG is lower than the RLS algorithm, while the complexity of the IDCCG solution depends on the number of iterations J.

3.4 Proposed Diffusion Distributed CG–Based Algorithms

In this section, we detail the proposed diffusion distributed CCG (DDCCG) and diffusion distributed MCG (DDMCG) algorithms for distributed parameter estimation and spectrum estimation using wireless sensor networks.

3.4.1 Diffusion Distributed CG–Based Algorithms

In the derivation of diffusion distributed CG–based strategy, we consider a network structure where each node from the same neighborhood could exchange information with each other at every time instant. For each node in the network, the CTA scheme [17] is employed. Each node can collect information from all its neighbors and itself, and then convey all the information to its local adaptive algorithm and update the estimate of the weight vector through our algorithms. Specifically, at any time instant i, we define that node k has access to a set of estimates $\{\boldsymbol{\omega}_l(i-1)\}_{l\in\mathcal{N}_k}$ from its neighbors, where $\mathcal{N}_k$ denotes the set of neighbor nodes of node k including node k itself. Then, these local estimates are combined at node k as

$$\boldsymbol{\psi}_k(i) = \sum_{l\in\mathcal{N}_k} c_{kl}\boldsymbol{\omega}_l(i-1) \tag{3.27}$$

where c_{kl} is calculated through the Metropolis rule in (2.16) due to its simplicity and good performance [55]. For the proposed diffusion distributed CG–based algorithms, the whole processing is shown in Fig. 3.2.

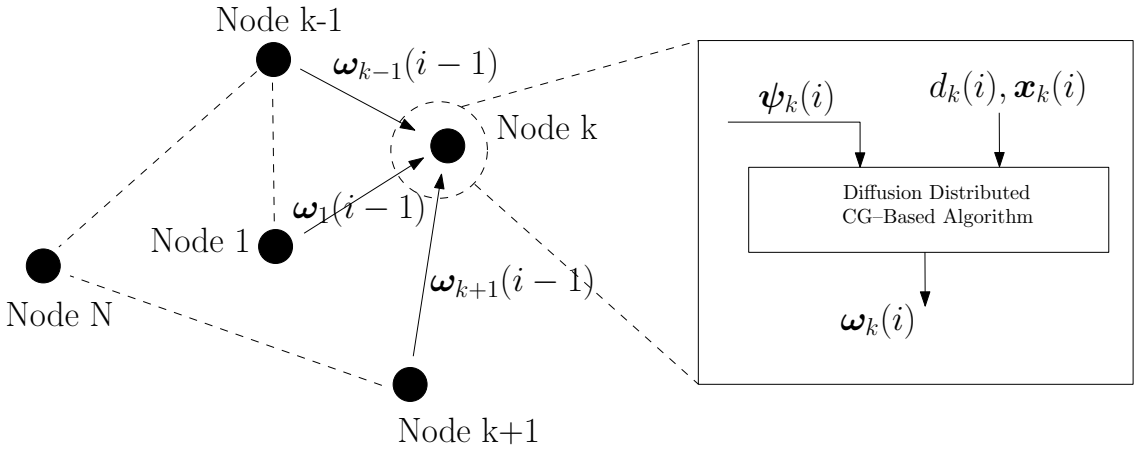

Figure 3.2: Diffusion Distributed CG–Based Network Processing

Proposed DDCCG Algorithm

For the DDCCG algorithm, (3.27) is employed to combine the estimates $\boldsymbol{\omega}_l(i-1), l \in \mathcal{N}_k$ from node k's neighbor nodes and then the estimate at node k is updated as follows:

$$\boldsymbol{\omega}_k^j(i) = \boldsymbol{\omega}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i), \tag{3.28}$$

where $\boldsymbol{\omega}_k^0(i) = \boldsymbol{\psi}_k(i)$. The rest of the derivation is similar to the IDCCG solution and the pseudo–code is detailed in Table 3.4.

Proposed DDMCG Algorithm

For the DDMCG algorithm, the iteration j is removed and the estimate at node k is updated as:

$$\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i) + \alpha_k(i)\boldsymbol{p}_k(i), \tag{3.29}$$

The complete DDMCG algorithm is described in Table 3.5.

3.4.2 Computational Complexity

The proposed diffusion distributed CG–based algorithms are analysed in terms of computational complexity, where additions and multiplications are measured. The details are listed in Table 3.6. Similarly to the incremental distributed CG–based algorithms, it is clear that the complexity of the DDCCG solution depends on the iteration number J and both DDCCG and DDMCG solutions depend on the number of neighbor nodes $|\mathcal{N}_k|$ of node k. The parameter M is the length of the unknown vector $\boldsymbol{\omega}_0$ that needs to be estimated.

3.5 Preconditioner Design

Preconditioning is an important technique which can be used to improve the performance of CG algorithms [56–59]. The idea behind preconditioning is to employ the CG algorithms on an equivalent system or in a transform–domain. Thus, instead of solving $\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{b}$ we solve a related problem $\tilde{\boldsymbol{R}}\tilde{\boldsymbol{\omega}} = \tilde{\boldsymbol{b}}$, which is modified with the aim of obtaining better convergence and steady state performance. The relationships between these two equations are given by

$$\tilde{\boldsymbol{R}} = \boldsymbol{T}\boldsymbol{R}\boldsymbol{T}^H, \tag{3.30}$$

Table 3.4: DDCCG Algorithm

Initialization:

$\boldsymbol{\omega}_k(0) = \mathbf{0}, k = 1, 2, \ldots, N$

For each time instant $i = 1, 2, \ldots, I$

For each node $k = 1, 2, \ldots, N$ (Combination Step)

$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\omega}_l(i-1)$

End

For each node $k = 1, 2, \ldots, N$ (Adaptation Step)

$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$

$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i)$

$\boldsymbol{\omega}_k^0(i) = \boldsymbol{\psi}_k(i)$

$\boldsymbol{p}_k^0(i) = \boldsymbol{g}_k^0(i) = \boldsymbol{b}_k(i) - \boldsymbol{R}_k(i)\boldsymbol{\omega}_k^0(i)$

For iterations $j = 1, 2, \ldots, J$

$$\alpha_k^j(i) = \frac{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}{\left(\boldsymbol{p}_k^{j-1}(i)\right)^H \boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)}$$

$\boldsymbol{\omega}_k^j(i) = \boldsymbol{\omega}_k^{j-1}(i) + \alpha_k^j(i)\boldsymbol{p}_k^{j-1}(i)$

$\boldsymbol{g}_k^j(i) = \boldsymbol{g}_k^{j-1}(i) - \alpha_k^j(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k^{j-1}(i)$

$$\beta_k^j(i) = \frac{\left(\boldsymbol{g}_k^j(i)\right)^H \boldsymbol{g}_k^j(i)}{\left(\boldsymbol{g}_k^{j-1}(i)\right)^H \boldsymbol{g}_k^{j-1}(i)}$$

$\boldsymbol{p}_k^j(i) = \boldsymbol{g}_k^j(i) + \beta_k^j(i)\boldsymbol{p}_k^{j-1}(i)$

End

$\boldsymbol{\omega}_k(i) = \boldsymbol{\omega}_k^J(i)$

End

End

Table 3.5: DDMCG Algorithm

Initialization:

$\boldsymbol{\omega}_k(0) = \mathbf{0}, k = 1, 2, \ldots, N$

For each node $k = 1, 2, \ldots, N$

$\boldsymbol{b}_k(1) = d_k^*(1)\boldsymbol{x}_k(1)$

$\boldsymbol{p}_k(0) = \boldsymbol{g}_k(0) = \boldsymbol{b}_k(1)$

End

For each time instant $i = 1, 2, \ldots, I$

For each node $k = 1, 2, \ldots, N$(Combination Step)

$\boldsymbol{\psi}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl}\boldsymbol{\omega}_l(i-1)$

End

For each node $k = 1, 2, \ldots, N$(Adaptation Step)

$\boldsymbol{R}_k(i) = \lambda_f \boldsymbol{R}_k(i-1) + \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$

$\boldsymbol{b}_k(i) = \lambda_f \boldsymbol{b}_k(i-1) + d_k^*(i)\boldsymbol{x}_k(i)$

$\alpha_k(i) = \eta \frac{\boldsymbol{p}_k^H(i-1)\boldsymbol{g}_k(i-1)}{\boldsymbol{p}_k^H(i-1)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1)}$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$

$\boldsymbol{\omega}_k(i) = \boldsymbol{\psi}_k(i) + \alpha_k(i)\boldsymbol{p}_k(i-1)$

$\boldsymbol{g}_k(i) = \lambda_f \boldsymbol{g}_k(i-1) - \alpha_k(i)\boldsymbol{R}_k(i)\boldsymbol{p}_k(i-1) + \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\psi}_{k-1}^H(i)\boldsymbol{x}_k(i)]$

$\beta_k(i) = \frac{\left(\boldsymbol{g}_k(i) - \boldsymbol{g}_k(i-1)\right)^H \boldsymbol{g}_k(i)}{\boldsymbol{g}_k^H(i-1)\boldsymbol{g}_k(i-1)}$

$\boldsymbol{p}_k(i) = \boldsymbol{g}_k(i) + \beta_k(i)\boldsymbol{p}_k(i-1)$

End

End

Table 3.6: Computational Complexity Of Different Diffusion Algorithms per Node

Algorithm	Additions	Multiplications
DDCCG	$2M^2 + M$ $+J(M^2 + 5M$ $+\lvert\mathcal{N}_k\rvert M - 4)$	$3M^2 + 2M$ $+J(M^2 + 6M$ $+\lvert\mathcal{N}_k\rvert M + 2)$
DDMCG	$2M^2 + 9M - 4$ $+\lvert\mathcal{N}_k\rvert M$	$3M^2 + 10M + 2$ $+\lvert\mathcal{N}_k\rvert M$
Diffusion LMS [17]	$4M - 1 + \lvert\mathcal{N}_k\rvert M$	$3M + 1 + \lvert\mathcal{N}_k\rvert M$
Diffusion RLS [20]	$4M^2 + 16M + 1 + \lvert\mathcal{N}_k\rvert M$	$4M^2 + 12M - 1 + \lvert\mathcal{N}_k\rvert M$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{T}\boldsymbol{\omega} \tag{3.31}$$

and

$$\tilde{\boldsymbol{b}} = \boldsymbol{T}\boldsymbol{b}, \tag{3.32}$$

where the $M \times M$ matrix $\boldsymbol{T}$ is called a preconditioner. We design the matrix $\boldsymbol{T}$ as an arbitrary unitary matrix that has the following property [10]

$$\boldsymbol{T}\boldsymbol{T}^H = \boldsymbol{T}^H\boldsymbol{T} = \boldsymbol{I}. \tag{3.33}$$

Two kinds of unitary transformations are considered to build the preconditioner $\boldsymbol{T}$, which are discrete Fourier transform (DFT) and discrete cosine transform (DCT) [10]. The motivation behind employing these two matrix is they have useful de–correlation properties and often reduce the eigenvalue spread of the auto–correlation matrix of the input signal [10].

For the DFT scheme, we employ the following expression

$$[\boldsymbol{T}_{DFT}]_{vm} \triangleq \frac{1}{\sqrt{M}} e^{-\frac{j2\pi mv}{M}}, \quad v, m = 0, 1, 2, \ldots, M-1, \tag{3.34}$$

where v indicates the row index and m the column index. M is the length of the unknown parameter $\boldsymbol{\omega}_0$. The matrix form of $\boldsymbol{T}_{DFT}$ is illustrated as

$$\boldsymbol{T}_{DFT} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{j2\pi}{M}} & e^{-\frac{j4\pi}{M}} & \cdots & e^{-\frac{j2(M-1)\pi}{M}} \\ 1 & e^{-\frac{j4\pi}{M}} & e^{-\frac{j8\pi}{M}} & \cdots & e^{-\frac{j4(M-1)\pi}{M}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{j2(M-1)\pi}{M}} & e^{-\frac{j4(M-1)\pi}{M}} & \cdots & e^{-\frac{j2(M-1)^2\pi}{M}} \end{bmatrix} \tag{3.35}$$

For the DCT scheme, the preconditioner $\boldsymbol{T}$ is defined as

$$[\boldsymbol{T}_{DCT}]_{vm} \triangleq \delta(v) \cos\left(\frac{v(2m+1)\pi}{2M}\right), \quad v, m = 0, 1, 2, \ldots, M-1, \tag{3.36}$$

where

$$\delta(0) = \frac{1}{\sqrt{M}} \text{ and } \delta(v) = \sqrt{\frac{2}{M}} \text{ for } v \neq 0 \tag{3.37}$$

and the matrix form of $\boldsymbol{T}_{DCT}$ is illustrated as

$$\boldsymbol{T}_{DCT} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \sqrt{2}\cos(\frac{3\pi}{2M}) & \sqrt{2}\cos(\frac{5\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{(2M-1)\pi}{2M}) \\ 1 & \sqrt{2}\cos(\frac{6\pi}{2M}) & \sqrt{2}\cos(\frac{10\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{2(2M-1)\pi}{2M}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \sqrt{2}\cos(\frac{3(M-1)\pi}{2M}) & \sqrt{2}\cos(\frac{5(M-1)\pi}{2M}) & \cdots & \sqrt{2}\cos(\frac{(2M-1)(M-1)\pi}{2M}) \end{bmatrix} \tag{3.38}$$

Then, for the DCT scheme, we choose $\boldsymbol{T} = \boldsymbol{T}_{DCT}^{H}$. It should be noticed that the scaling factor $\frac{1}{\sqrt{M}}$ is added in the expression for the $\boldsymbol{T}_{DFT}$ in order to result in a unitary transformation since then $\boldsymbol{T}_{DFT}$ satisfies $\boldsymbol{T}_{DFT}\boldsymbol{T}_{DFT}^{H} = \boldsymbol{T}_{DFT}^{H}\boldsymbol{T}_{DFT} = \boldsymbol{I}$ [10].

The optimal selection of the preconditioner is the Karhunen–Loève transform (KLT) [10]. However, using the KLT is not practical since it requires knowledge of the auto–correlation matrix $\boldsymbol{R}$ of the input signal and this information is generally lacking in implementations.

3.6 Simulation Results

In this section, we investigate the performance of the proposed incremental and diffusion distributed CG–based algorithms in two scenarios: distributed estimation and distributed spectrum estimation in wireless sensor networks.

3.6.1 Distributed Estimation in Wireless Sensor Networks

In this subsection, we compare the proposed incremental and diffusion distributed CG–based algorithms with LMS [1, 17] and RLS [1, 20] algorithms, based on the MSE and MSD performance metrics. For each comparison, the number of time instants is set to 1000, and we assume there are 20 nodes in the network. The length of the unknown parameter vector $\boldsymbol{\omega}_0$ is 10, the variances for the input signal and the noise are 1 and 0.001, respectively. In addition, the noise samples are modeled as circular Gaussian noise with zero mean.

Performance of Proposed Incremental Distributed CG–Based Algorithms

First, we define the parameters of the performance test for each algorithm and the network. The step size μ for the LMS algorithm [1] is set to 0.2, the forgetting factor λ for the RLS [1] algorithm is set to 0.998. The λ_f for IDCCG and IDMCG are both set to 0.998. For IDMCG, the η is equal to 0.55. The iteration number J for IDCCG is set to 5. We choose the DCT matrix as the preconditioner.

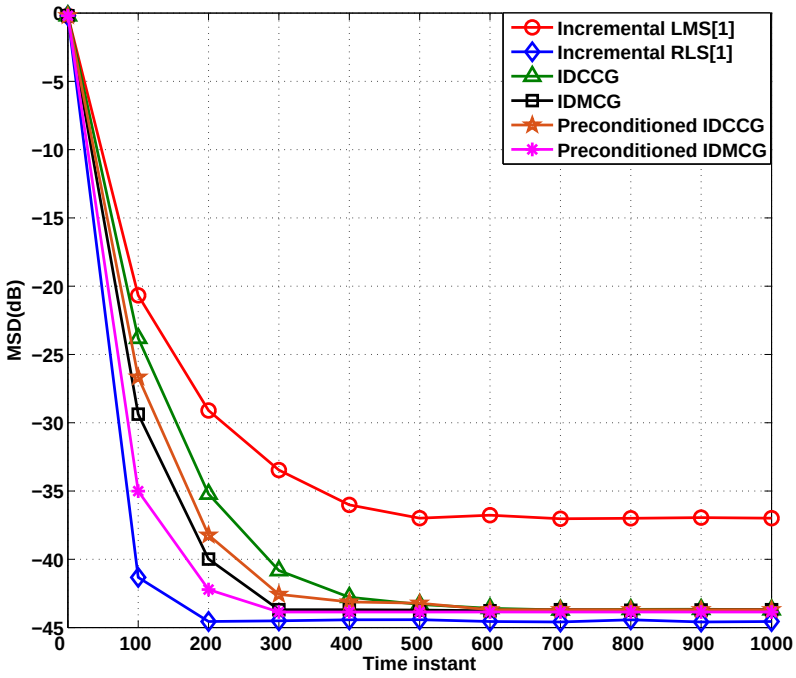

Figure 3.3: MSD performance comparison for the incremental distributed strategies

The MSD and MSE performances of each algorithm is shown in Fig. 3.3 and 3.4 respectively. We can verify that, the IDMCG and IDCCG algorithm performs better than incremental LMS, while IDMCG is close to the RLS algorithm. With the preconditioning strategy, the performance of the IDCCG and IDMCG is further improved. The reason why the proposed IDMCG algorithm has a better performance than IDCCG is because IDMCG employs the negative gradient vector $\boldsymbol{g}_k$ with a recursive expression and the β_k is computed using the Polak–Ribiere approach, which results in more accurate estimates. Comparing with the IDCCG algorithm, the IDMCG is a non–reset and low complexity algorithm with one iteration per time instant. Since the frequency which the algorithm resets influences the performance, the IDMCG algorithm introduces the non–reset method

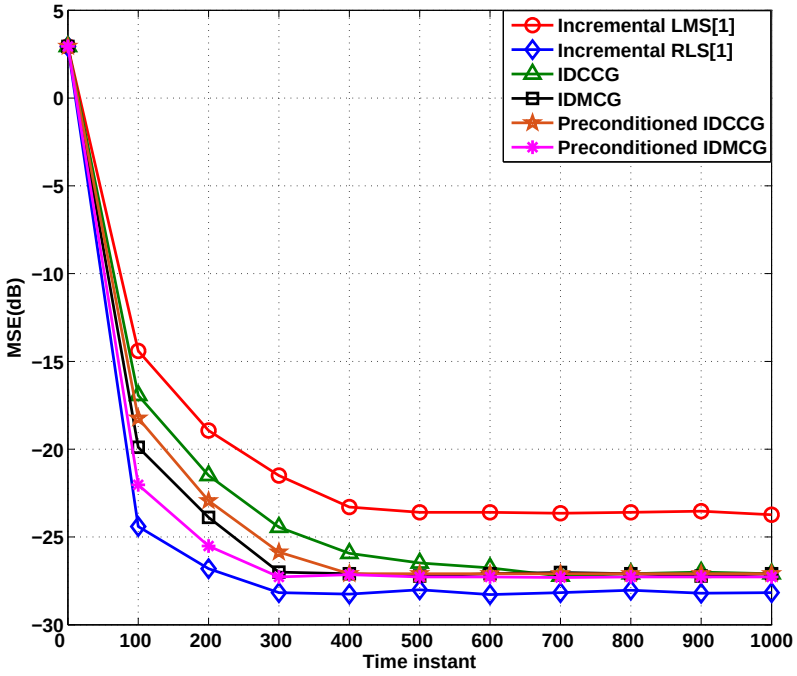

Figure 3.4: MSE performance comparison for the incremental distributed strategies

together with the Polak– Ribiere approach which are used to improve the performance [8]. In detail, according to (3.26), when $\boldsymbol{g}_k(i) \approx \boldsymbol{g}_k(i-1)$, $\beta_k(i) \approx 0$. This will lead to $\boldsymbol{p}_k(i) \approx \boldsymbol{g}_k(i)$, which means the algorithm is reset.

Performance of Proposed Diffusion Distributed CG–Based Algorithms

The parameters of the performance test for each algorithm and the network are defined as follows: the step size μ for the LMS [17] algorithm is set to 0.2, the forgetting factor λ for the RLS [20] algorithm is set to 0.998. The λ_f for DDCCG and DDMCG are both 0.998. The η is equal to 0.45 for DDMCG. The iteration number J for DDCCG is set to 5. We choose the DCT matrix as the preconditioner.

For the diffusion strategy, the combining coefficients c_{kl} are calculated following the Metropolis rule. Fig. 3.5 shows the network structure. The results are illustrated in Fig. 3.6 and 3.7. We can see that, the proposed DDMCG and DDCCG still have a better performance than the LMS algorithm and DDMCG is closer to the RLS's performance. The performance of the DDCCG and DDMCG can still benefit from the preconditioning

strategy.

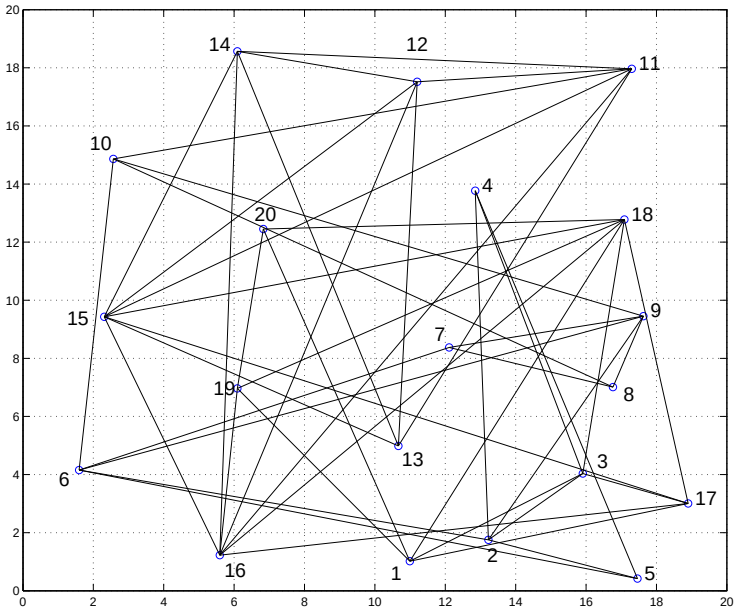

Figure 3.5: Network structure

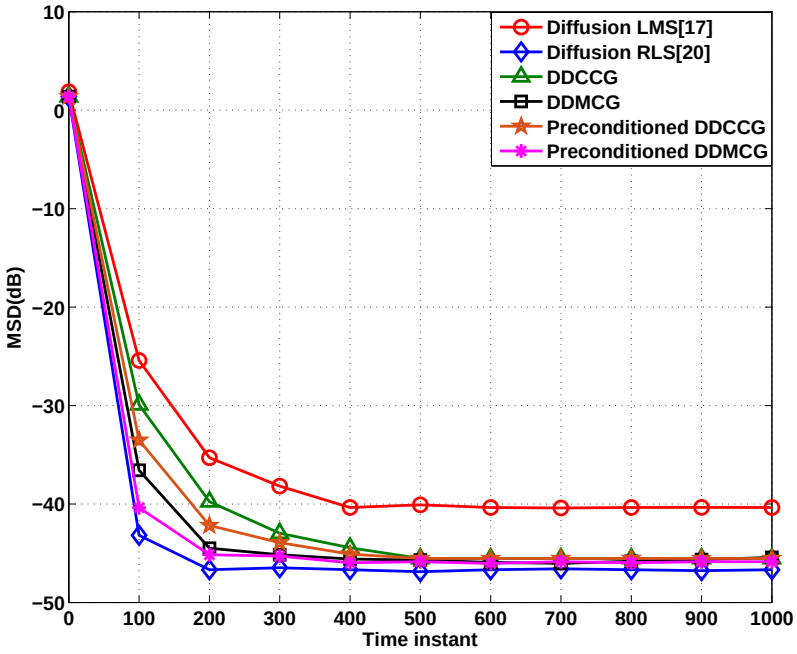

Figure 3.6: MSD performance comparison for the diffusion distributed strategies

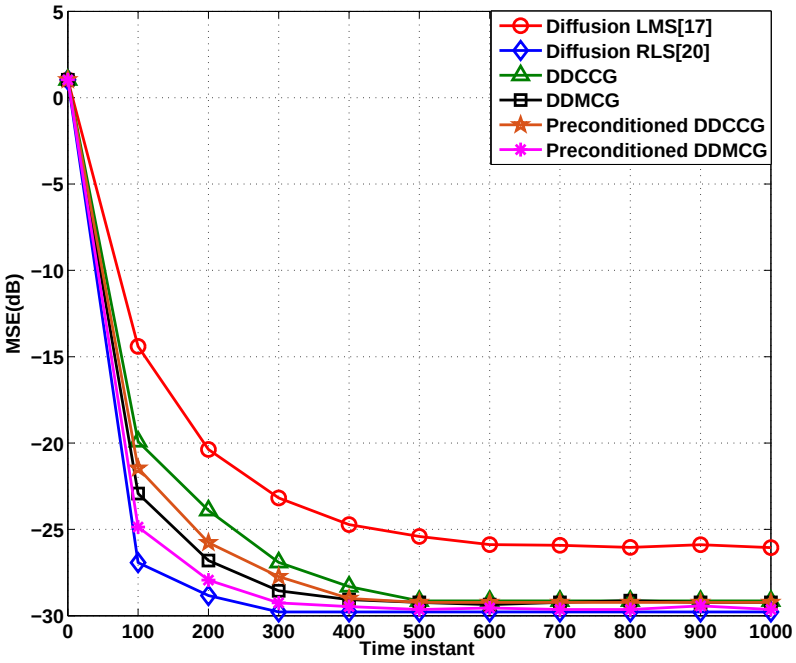

Figure 3.7: MSE performance comparison for the diffusion distributed strategies

3.6.2 Distributed Spectrum Estimation

In this simulation, we consider a network composed of $N = 20$ nodes estimating the unknown spectrum $\boldsymbol{\omega}_0$, as illustrated in Fig. 3.5. The nodes scan $N_c = 100$ frequencies over the frequency axis, which is normalized between 0 and 1, and use $\mathcal{B} = 50$ non–overlapping rectangular basis functions to model the expansion of the spectrum [25]. The basis functions have amplitude equal to one. We assume that the unknown spectrum $\boldsymbol{\omega}_0$ is transmitted over 8 basis functions, thus leading to a sparsity ratio equal to 8/50. The power transmitted over each basis function is set equal to 0.7. The variance for the observation noise is 0.01.

For distributed estimation, we employ the DDMCG and the DDCCG algorithms, together with the preconditioned DDMCG algorithm to solve the cost function (3.11). The λ_f for DDCCG and DDMCG are both 0.99. The η_f is equal to 0.3 for DDMCG. The iteration number J for DDCCG is set to 5. The DCT matrix is employed as the preconditioner. We compare the proposed DDCCG and DDMCG algorithms with the sparse ATC diffusion algorithm [25], diffusion LMS algorithm [17] and diffusion RLS algorithm [20].

The step–sizes for the sparse ATC diffusion algorithm and diffusion LMS algorithm are set equal to 0.2, while for the sparse ATC diffusion algorithm, γ is set to 2.2×10^{-3} and β is set to 50. The forgetting factor λ for the diffusion RLS algorithm is set to 0.998.

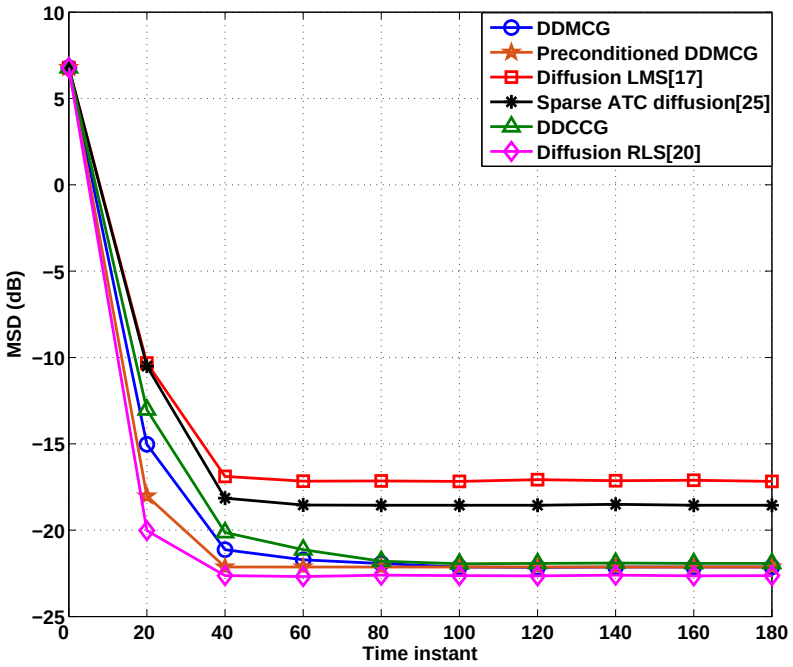

Figure 3.8: Performance comparison for the distributed spectrum estimation

We illustrate the result of distributed spectrum estimation carried out by different algorithms in the term of the MSD comparison in Fig. 3.8. We also select the sparse ATC diffusion algorithm [25], diffusion LMS algorithm [17] and DDMCG to compare their performance in term of PSD in Fig. 3.9. The true transmitted spectrum is also reported in Fig. 3.9.

From Fig. 3.8, the DDMCG still performs better than other algorithms and is close to the diffusion RLS algorithm. From Fig. 3.9, we can notice that all the algorithms are able to identify the spectrum, but it is also clear that the DDMCG algorithm is able to strongly reduce the effect of the spurious terms.

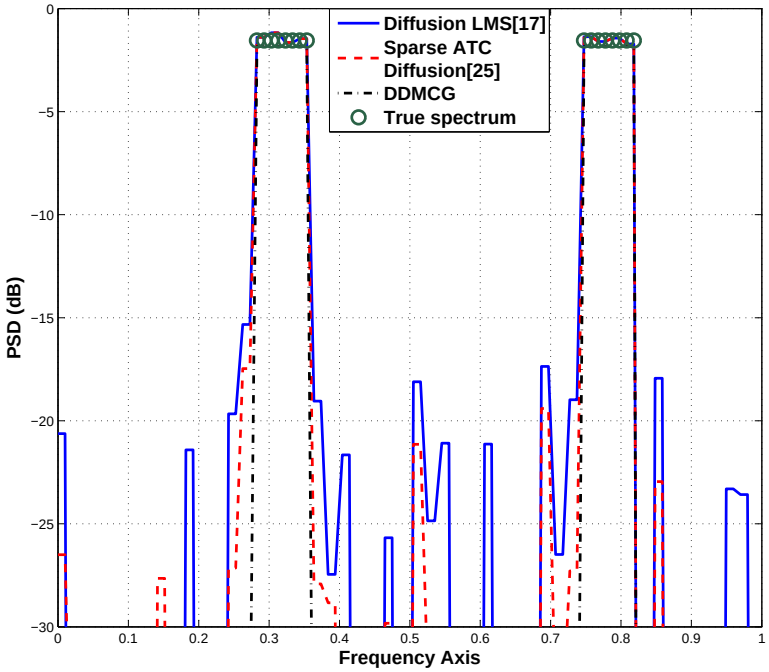

Figure 3.9: Example of distributed spectrum estimation

3.7 Summary

In this chapter, we have proposed distributed CG algorithms for both incremental and diffusion adaptive strategies. We have investigated the proposed algorithms in distributed estimation for wireless sensor networks and distributed spectrum estimation. The CG–based strategies have a lower computational complexity than RLS–type algorithms, depending on the number of iterations that CG employs and a faster convergence than the LMS algorithm. The preconditioning strategy is also introduced to further improve the performance of the proposed algorithms. Simulation results have illustrated the advantages of the proposed IDCCG/IDMCG and DDCCG/DDMCG algorithms in different applications.

Chapter 4

Adaptive Link Selection Algorithms for Distributed Diffusion Estimation

Contents

4.1 Introduction 59

4.2 System Model and Problem Statement 62

4.3 Proposed Adaptive Link Selection Algorithms 64

4.4 Analysis of the proposed algorithms 72

4.5 Simulations 90

4.6 Summary 100

4.1 Introduction

Distributed signal processing algorithms have become a key approach for statistical inference in wireless networks and applications such as wireless sensor networks and smart grids [1–4]. It is well known that distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area [1]. In this context, for each specific node, a set of neighbor nodes collect their local information and transmit the estimates to a specific node. Then, each specific node combines the collected information together with its local estimate to generate an improved

estimate.

4.1.1 Prior and Related Work

Several works in the literature have proposed strategies for distributed processing which include incremental [1, 60–62], diffusion [2, 17], sparsity–aware [3, 63] and consensus–based strategies [18]. With the incremental strategy, the processing follows a Hamiltonian cycle, i.e., the information flows through these nodes in one direction, which means each node passes the information to its adjacent node in a uniform direction. However, in order to determine an optimum cyclic path that covers all nodes (considering the noise, interference, path loss and channels between neighbor nodes), this method needs to solve an NP–hard problem. In addition, when any of the nodes fails, data communication through the cycle is interrupted and the distributed processing breaks down [1].

In distributed diffusion strategies [2, 63], the neighbors for each node are fixed and the combining coefficients are calculated after the network topology is deployed and starts its operation. One potential risk of this approach is that the estimation procedure may be affected by poorly performing links. More specifically, the fixed neighbors and the pre–calculated combining coefficients may not provide an optimized estimation performance for each specified node because there are links that are more severely affected by noise or fading. Moreover, when the number of neighbor nodes is large, each node requires a large bandwidth and transmit power. Prior work on topology design and adjustment techniques includes the studies in [64, 65] and [66], which are not dynamic in the sense that they cannot track changes in the network and mitigate the effects of poor links.

4.1.2 Contributions

The adaptive link selection algorithms for distributed estimation problems are proposed and studied in this chapter. Specifically, we develop adaptive link selection algorithms that can exploit the knowledge of poor links by selecting a subset of data from neighbor nodes. The first approach consists of exhaustive search–based LMS/RLS link selection (ES–LMS/ES–RLS) algorithms, whereas the second technique is based on sparsity–

inspired LMS/RLS link selection (SI–LMS/SI–RLS) algorithms. With both approaches, distributed processing can be divided into two steps. The first step is called the adaptation step, in which each node employs LMS or RLS to perform the adaptation through its local information. Following the adaptation step, each node will combine its collected estimates from its neighbors and local estimate, through the proposed adaptive link selection algorithms. The proposed algorithms result in improved estimation performance in terms of the mean–square error (MSE) associated with the estimates. In contrast to previously reported techniques, a key feature of the proposed algorithms is that the combination step involves only a subset of the data associated with the best performing links.

In the ES–LMS and ES–RLS algorithms, we consider all possible combinations for each node with its neighbors and choose the combination associated with the smallest MSE value. In the SI–LMS and SI–RLS algorithms, we incorporate a reweighted zero attraction (RZA) strategy into the adaptive link selection algorithms. The RZA approach is often employed in applications dealing with sparse systems in such a way that it shrinks the small values in the parameter vector to zero, which results in better convergence and steady–state performance. Unlike prior work with sparsity–aware algorithms [3, 13, 67, 68], the proposed SI–LMS and SI–RLS algorithms exploit the possible sparsity of the MSE values associated with each of the links in a different way. In contrast to existing methods that shrink the signal samples to zero, SI–LMS and SI–RLS shrink to zero the links that have poor performance or high MSE values. By using the SI–LMS and SI–RLS algorithms, data associated with unsatisfactory performance will be discarded, which means the effective network topology used in the estimation procedure will change as well. Although the physical topology is not changed by the proposed algorithms, the choice of the data coming from the neighbor nodes for each node is dynamic, leads to the change of combination weights and results in improved performance. We also remark that the topology could be altered with the aid of the proposed algorithms and a feedback channel which could inform the nodes whether they should be switched off or not. The proposed algorithms are considered for wireless sensor networks and also as a tool for distributed state estimation that could be used in smart grids.

In summary, the main contributions of this chapter are:

- We present adaptive link selection algorithms for distributed estimation that are able

to achieve significantly better performance than existing algorithms.

- We devise distributed LMS and RLS algorithms with link selection capabilities to perform distributed estimation.
- We analyze the MSE convergence and tracking performance of the proposed algorithms and their computational complexities and we derive analytical formulas to predict their MSE performance.
- A simulation study of the proposed and existing distributed estimation algorithms is conducted along with applications in wireless sensor networks and smart grids.

4.2 System Model and Problem Statement

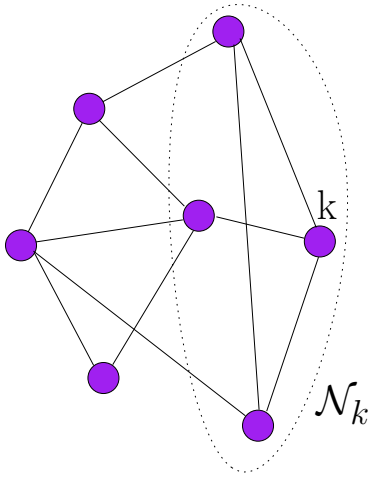

Figure 4.1: Network topology with N nodes

We consider a set of N nodes, which have limited processing capabilities, distributed over a given geographical area as depicted in Fig. 4.1. The nodes are connected and form a network, which is assumed to be partially connected because nodes can exchange information only with neighbors determined by the connectivity topology. We call a network with this property a partially connected network whereas a fully connected network means that data broadcast by a node can be captured by all other nodes in the network in one hop [19]. We can think of this network as a wireless network, but our analysis also applies to wired networks such as power grids. In our work, in order to perform link selection strategies, we assume that each node has at least two neighbors.

The aim of the network is to estimate an unknown parameter vector $\boldsymbol{\omega}_0$, which has length M. At every time instant i, each node k takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{4.1}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ random regression input signal vector and $n_k(i)$ denotes the Gaussian noise at each node with zero mean and variance $\sigma_{n,k}^2$. This linear model is able to capture or approximate well many input-output relations for estimation purposes [9] and we assume $I > M$. To compute an estimate of $\boldsymbol{\omega_0}$ in a distributed fashion, we need each node to minimize the MSE cost function [2]

$$J_k\big(\boldsymbol{\omega}_k(i)\big) = \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big|^2, \tag{4.2}$$

where $\mathbb{E}$ denotes expectation and $\boldsymbol{\omega}_k(i)$ is the estimated vector generated by node k at time instant i. Equation (4.3) is also the definition of the MSE and the global network cost function could be described as

$$J(\boldsymbol{\omega}) = \sum_{k=1}^{N} \mathbb{E}|d_k(i) - \boldsymbol{\omega}^H \boldsymbol{x}_k(i)|^2. \tag{4.3}$$

To solve this problem, diffusion strategies have been proposed in [2, 17] and [16]. In these strategies, the estimate for each node is generated through a fixed combination strategy given by

$$\boldsymbol{\omega}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\psi}_l(i), \tag{4.4}$$

where $\mathcal{N}_k$ denotes the set of neighbors of node k including node k itself, $c_{kl} \geq 0$ is the combining coefficient and $\boldsymbol{\psi}_l(i)$ is the local estimate generated by node l through its local information.

There are many ways to calculate the combining coefficient c_{kl} which include the Hastings [55], the Metropolis [30], the Laplacian [31] and the nearest neighbor [32] rules. In this work, due to its simplicity and good performance we adopt the Metropolis rule [30] given by (2.16).

The set of coefficients c_{kl} should satisfy [2]

$$\sum_{l \in \mathcal{N}_k \ \forall k} c_{kl} = 1. \tag{4.5}$$

For the combination strategy mentioned in (4.4), the choice of neighbors for each node is fixed, which results in some problems and limitations, namely:

- Some nodes may face high levels of noise or interference, which may lead to inaccurate estimates.
- When the number of neighbors for each node is high, large communication bandwidth and high transmit power are required.
- Some nodes may shut down or collapse due to network problems. As a result, local estimates to their neighbors may be affected.

Under such circumstances, a performance degradation is likely to occur when the network cannot discard the contribution of poorly performing links and their associated data in the estimation procedure. In the next section, the proposed adaptive link selection algorithms are presented, which equip a network with the ability to improve the estimation procedure. In the proposed scheme, each node is able to dynamically select the data coming from its neighbors in order to optimize the performance of distributed estimation techniques.

4.3 Proposed Adaptive Link Selection Algorithms

In this section, we present the proposed adaptive link selection algorithms. The goal of the proposed algorithms is to optimize the distributed estimation and improve the performance of the network by dynamically changing the topology. These algorithmic strategies give the nodes the ability to choose their neighbors based on their MSE performance. We develop two categories of adaptive link selection algorithms; the first one is based on an exhaustive search, while the second is based on a sparsity–inspired relaxation. The details will be illustrated in the following subsections.

4.3.1 Exhaustive Search–Based LMS/RLS Link Selection

The proposed ES–LMS and ES–RLS algorithms employ an exhaustive search to select the links that yield the best performance in terms of MSE. First, we describe how we define the adaptation step for these two strategies. In the ES–LMS algorithm, we employ the adaptation strategy given by

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big]^*, \tag{4.6}$$

where μ_k is the step size for each node. In the ES–RLS algorithm, we employ the following steps for the adaptation:

$$\begin{aligned}\boldsymbol{\Phi}_k^{-1}(i) &= \lambda^{-1}\boldsymbol{\Phi}_k^{-1}(i-1) \\ &\quad - \frac{\lambda^{-2}\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}_k^{-1}(i-1)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)},\end{aligned} \tag{4.7}$$

where λ is the forgetting factor. Then, we let

$$\boldsymbol{P}_k(i) = \boldsymbol{\Phi}_k^{-1}(i) \tag{4.8}$$

and

$$\boldsymbol{k}_k(i) = \frac{\lambda^{-1}\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}. \tag{4.9}$$

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \boldsymbol{k}(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big]^*, \tag{4.10}$$

$$\boldsymbol{P}_k(i+1) = \lambda^{-1}\boldsymbol{P}_k(i) - \lambda^{-1}\boldsymbol{k}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i). \tag{4.11}$$

Following the adaptation step, we introduce the combination step for both ES–LMS and ES–RLS algorithms, based on an exhaustive search strategy. At first, we introduce a tentative set Ω_k using a combinatorial approach described by

$$\Omega_k \in 2^{|\mathcal{N}_k|} \backslash \varnothing, \tag{4.12}$$

where the set Ω_k is a nonempty set with $2^{|\mathcal{N}_k|}$ elements. After the tentative set Ω_k is defined, we write the cost function (4.2) for each node as

$$J_k\big(\boldsymbol{\psi}(i)\big) \triangleq \mathbb{E}\big|d_k(i) - \boldsymbol{\psi}^H(i)\boldsymbol{x}_k(i)\big|^2, \tag{4.13}$$

where

$$\boldsymbol{\psi}(i) \triangleq \sum_{l\in\Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i) \tag{4.14}$$

is the local estimator and $\boldsymbol{\psi}_l(i)$ is calculated through (4.6) or (4.10), depending on the algorithm, i.e., ES–LMS or ES–RLS. With different choices of the set Ω_k, the combining coefficients c_{kl} will be re–calculated through (2.16), to ensure condition (4.5) is satisfied.

Then, we introduce the error pattern for each node, which is defined as

$$e_{\Omega_k}(i) \triangleq d_k(i) - \left[\sum_{l\in\Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)\right]^H \boldsymbol{x}_k(i). \tag{4.15}$$

For each node k, the strategy that finds the best set $\Omega_k(i)$ must solve the following optimization problem:

$$\hat{\Omega}_k(i) = \arg \min_{\Omega_k \in 2^{N_k}\setminus\varnothing} |e_{\Omega_k}(i)|. \tag{4.16}$$

After all steps have been completed, the combination step in (4.4) is performed as described by

$$\boldsymbol{\omega}_k(i+1) = \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i). \tag{4.17}$$

At this stage, the main steps of the ES–LMS and ES–RLS algorithms have been completed. The proposed ES–LMS and ES–RLS algorithms find the set $\widehat{\Omega}_k(i)$ that minimizes the error pattern in (4.15) and (4.16) and then use this set of nodes to obtain $\boldsymbol{\omega}_k(i)$ through (4.17). The ES–LMS/ES–RLS algorithms are briefly summarized as follows:

Step 1 Each node performs the adaptation through its local information based on the LMS or RLS algorithm.

Step 2 Each node finds the best set $\Omega_k(i)$, which satisfies (4.16).

Step 3 Each node combines the information obtained from its best set of neighbors through (4.17).

The details of the proposed ES–LMS and ES–RLS algorithms are shown in Tables 4.1 and 4.2. When the ES–LMS and ES–RLS algorithms are implemented in networks with a large number of small and low–power sensors, the computational complexity cost may become high, as the algorithm in (4.16) requires an exhaustive search and needs more computations to examine all the possible sets $\Omega_k(i)$ at each time instant.

Table 4.1: The ES-LMS Algorithm

Initialize: $\boldsymbol{\omega}_k(1)$=0, for $k = 1, 2, \ldots, N$

For each time instant $i = 1, 2, ..., I$

For each node $k = 1, 2, \ldots, N$

$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*$

end

For each node $k = 1, 2, \ldots, N$

find all possible sets of Ω_k

$e_{\Omega_k}(i) = d_k(i) - [\sum_{l \in \Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)]^H \boldsymbol{x}_k(i)$

$\widehat{\Omega}_k(i) = \arg\min_{\Omega_k} |e_{\Omega_k}(i)|$

$\boldsymbol{\omega}_k(i+1) = \sum_{l \in \widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i)$

end

end

4.3.2 Sparsity–Inspired LMS/RLS Link Selection

The ES–LMS/ES–RLS algorithms previously outlined need to examine all possible sets to find a solution at each time instant, which might result in high computational complexity for large networks operating in time–varying scenarios. To solve the combinatorial problem with reduced complexity, we propose sparsity-inspired based SI–LMS and SI–RLS algorithms, which are as simple as standard diffusion LMS or RLS algorithms and are suitable for adaptive implementations and scenarios where the parameters to be estimated are slowly time–varying. The zero–attracting strategy (ZA), reweighted zero–attracting strategy (RZA) and zero–forcing (ZF) are reported in [3] and [69] as for sparsity aware techniques. These approaches are usually employed in applications dealing with sparse systems in scenarios where they shrink the small values in the parameter vector to zero, which results in better convergence rate and steady–state performance. Unlike existing methods that shrink the signal samples to zero, the proposed SI–LMS and SI–RLS algorithms shrink to zero the links that have poor performance or high MSE values. To detail the novelty of the proposed sparsity–inspired LMS/RLS link selection algorithms, we illustrate the processing in Fig.4.2.

Table 4.2: The ES-RLS Algorithm

Initialize: $\boldsymbol{\omega}_k(1)$=0, for $k = 1, 2, \ldots, N$

$\boldsymbol{\Phi}_k^{-1}(0) = \delta^{-1}\boldsymbol{I}, \delta =$ small positive constant

For each time instant $i = 1, 2, ..., I$

For each node $k = 1, 2, \ldots, N$

$$\boldsymbol{\Phi}_k^{-1}(i) = \lambda^{-1}\boldsymbol{\Phi}_k^{-1}(i-1) - \frac{\lambda^{-2}\boldsymbol{\Phi}_k^{-1}(i-1)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}^{-1}(i-1)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{\Phi}^{-1}(i-1)\boldsymbol{x}_k(i)}$$

$$\boldsymbol{P}_k(i) = \boldsymbol{\Phi}_k^{-1}(i)$$

$$\boldsymbol{k}_k(i) = \frac{\lambda^{-1}\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)}$$

$$\boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \boldsymbol{k}(i)[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)]^*$$

$$\boldsymbol{P}_k(i+1) = \lambda^{-1}\boldsymbol{P}_k(i) - \lambda^{-1}\boldsymbol{k}(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i)$$

end

For each node $k = 1, 2, \ldots, N$

find all possible sets of Ω_k

$$e_{\Omega_k}(i) = d_k(i) - [\sum_{l\in\Omega_k} c_{kl}(i)\boldsymbol{\psi}_l(i)]^H\boldsymbol{x}_k(i)$$

$$\widehat{\Omega}_k(i) = \arg\min_{\Omega_k} |e_{\Omega_k}(i)|$$

$$\boldsymbol{\omega}_k(i+1) = \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i)$$

end

end

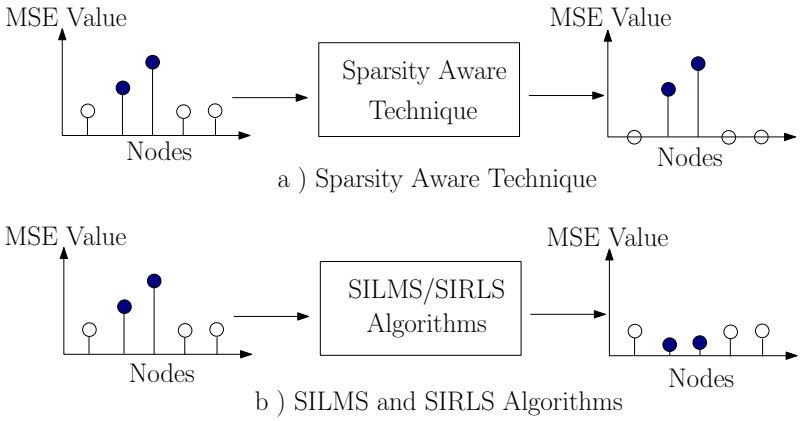

Figure 4.2: Sparsity aware signal processing strategies

Fig. 4.2 (a) shows a standard type of sparsity–aware processing. We can see that, after being processed by a sparsity–aware algorithm, the nodes with small MSE values will be shrunk to zero. In contrast, the proposed SI–LMS and SI–RLS algorithms will keep the nodes with lower MSE values and reduce the combining weight of the nodes with large MSE values as illustrated in Fig. 4.2 (b). When compared with ES–type algorithms, the SI–LMS/RLS algorithms do not need to consider all possible combinations of nodes, which means the SI–LMS/RLS algorithms have lower complexity. In the following, we will show how the proposed SI–LMS/SI–RLS algorithms are employed to realize the link selection strategy automatically.

In the adaptation step, we follow the same procedure in (4.6)–(4.10) as that of the ES–LMS and ES–RLS algorithms for the SI–LMS and SI–RLS algorithms, respectively. Then we reformulate the combination step. First, we introduce the log–sum penalty into the combination step in (4.4). Different penalty terms have been considered for this task. We have adopted a heuristic approach [3, 70] known as reweighted zero–attracting strategy into the combination step in (4.4) because this strategy has shown an excellent performance and is simple to implement. The log–sum penalty is defined as:

$$f_1(e_k(i)) = \sum_{l \in \mathcal{N}_k} \log\left(1 + \varepsilon |e_{kl}(i)|\right), \tag{4.18}$$

where the error $e_{kl}(i)(l \in \mathcal{N}_k)$, which stands for the neighbor node l of node k including node k itself, is defined as

$$e_{kl}(i) \triangleq d_k(i) - \boldsymbol{\psi}_l^H(i)\boldsymbol{x}_k(i) \tag{4.19}$$

and ε is the shrinkage magnitude. Then, we introduce the vector and matrix quantities

required to describe the combination step. We first define a vector $\boldsymbol{c}_k$ that contains the combining coefficients for each neighbor of node k including node k itself as described by

$$\boldsymbol{c}_k \triangleq [c_{kl}], \quad l \in \mathcal{N}_k. \tag{4.20}$$

Then, we define a matrix $\boldsymbol{\Psi}_k$ that includes all the estimated vectors, which are generated after the adaptation step of SI–LMS and of SI–RLS for each neighbor of node k including node k itself as given by

$$\boldsymbol{\Psi}_k \triangleq [\boldsymbol{\psi}_l(i)], \quad l \in \mathcal{N}_k. \tag{4.21}$$

Note that the adaptation steps of SI–LMS and SI–RLS are identical to (4.6) and (4.10), respectively. An error vector $\hat{\boldsymbol{e}}_k$ that contains all error values calculated through (4.19) for each neighbor of node k including node k itself is expressed by

$$\hat{\boldsymbol{e}}_k \triangleq [e_{kl}(i)], \quad l \in \mathcal{N}_k. \tag{4.22}$$

To devise the sparsity–inspired approach, we have modified the vector $\hat{\boldsymbol{e}}_k$ in the following way:

1. The element with largest absolute value $|e_{kl}(i)|$ in $\hat{\boldsymbol{e}}_k$ will be kept as $|e_{kl}(i)|$.
2. The element with smallest absolute value will be set to $-|e_{kl}(i)|$. This process will ensure the node with smallest error pattern has a reward on its combining coefficient.
3. The remaining entries will be set to zero.

At this point, the combination step can be defined as [70]

$$\boldsymbol{\omega}_k(i) = \sum_{j=1}^{|\mathcal{N}_k|} \left[c_{k,j} - \rho \frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} \right] \boldsymbol{\psi}_{k,j}, \tag{4.23}$$

where $c_{k,j}, \hat{e}_{k,j}$ stand for the jth element in the $\boldsymbol{c}_k, \hat{\boldsymbol{e}}_k$ and $\boldsymbol{\psi}_{k,j}$ stands for the jth column in $\boldsymbol{\Psi}_k$. The parameter ρ is used to control the algorithm's shrinkage intensity. We then calculate the partial derivative of $\hat{\boldsymbol{e}}_k[j]$:

$$\begin{aligned} \frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} &= \frac{\partial\big(\log(1+\varepsilon|e_{kl}(i)|)\big)}{\partial\big(e_{kl}(i)\big)} \\ &= \varepsilon \frac{\operatorname{sign}(e_{kl}(i))}{1+\varepsilon|e_{kl}(i)|} \quad l \in \mathcal{N}_k \\ &= \varepsilon \frac{\operatorname{sign}(\hat{e}_{k,j})}{1+\varepsilon|\hat{e}_{k,j}|}. \end{aligned} \tag{4.24}$$

To ensure that $\sum_{j=1}^{|\mathcal{N}_k|}\left(c_{k,j}-\rho\frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}}\right)=1$, we replace $\hat{e}_{k,j}$ with $\xi_{\min}$ in the denominator of (4.24), where the parameter $\xi_{\min}$ stands for the minimum absolute value of $e_{kl}(i)$ in $\hat{\boldsymbol{e}}_k$. Then, (4.24) can be rewritten as

$$\frac{\partial f_1(\hat{e}_{k,j})}{\partial \hat{e}_{k,j}} \approx \varepsilon \frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}. \tag{4.25}$$

At this stage, the log-sum penalty performs shrinkage and selects the set of estimates from the neighbor nodes with the best performance, at the combination step. The function $\text{sign}(a)$ is defined as

$$\text{sign}(a)=\begin{cases} a/|a| & a\neq 0 \\ 0 & a=0. \end{cases} \tag{4.26}$$

Then, by inserting (4.25) into (4.23), the proposed combination step is given by

$$\boldsymbol{\omega}_k(i)=\sum_{j=1}^{|\mathcal{N}_k|}\left[c_{k,j}-\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}\right]\boldsymbol{\psi}_{k,j}. \tag{4.27}$$

Note that the condition $c_{k,j}-\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}\geq 0$ is enforced in (4.27). When $c_{k,j}-\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}=0$, it means that the corresponding node has been discarded from the combination step. In the following time instant, if this node still has the largest error, there will be no changes in the combining coefficients for this set of nodes.

To guarantee the stability, the parameter ρ is assumed to be sufficiently small and the penalty takes effect only on the element in $\hat{\boldsymbol{e}}_k$ for which the magnitude is comparable to $1/\varepsilon$ [3]. Moreover, there is little shrinkage exerted on the element in $\hat{\boldsymbol{e}}_k$ whose $|\hat{\boldsymbol{e}}_k[j]| \ll 1/\varepsilon$. The SI–LMS and SI–RLS algorithms perform link selection by the adjustment of the combining coefficients through (4.27). At this point, it should be emphasized that:

- The process in (4.27) satisfies condition (4.5), as the penalty and reward amounts of the combining coefficients are the same for the nodes with maximum and minimum error, respectively, and there are no changes for the rest nodes in the set.
- When computing (4.27), there are no matrix–vector multiplications. Therefore, no additional complexity is introduced. As described in (4.23), only the jth element of $\boldsymbol{c}_k, \hat{\boldsymbol{e}}_k$ and jth column of $\boldsymbol{\Psi}_k$ are used for calculation.

For the neighbor node with the largest MSE value, after the modifications of $\hat{\boldsymbol{e}}_k$, its $e_{kl}(i)$ value in $\hat{\boldsymbol{e}}_k$ will be a positive number which will lead to the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (4.27)

being positive too. This means that the combining coefficient for this node will be shrunk and the weight for this node to build $\boldsymbol{\omega}_k(i)$ will be shrunk too. In other words, when a node encounters high noise or interference levels, the corresponding MSE value might be large. As a result, we need to reduce the contribution of that node.

In contrast, for the neighbor node with the smallest MSE, as its $e_{kl}(i)$ value in $\hat{\boldsymbol{e}}_k$ will be a negative number, the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (4.27) will be negative too. As a result, the weight for this node associated with the smallest MSE to build $\boldsymbol{\omega}_k(i)$ will be increased. For the remaining neighbor nodes, the entry $e_{kl}(i)$ in $\hat{\boldsymbol{e}}_k$ is zero, which means the term $\rho\varepsilon\frac{\text{sign}(\hat{e}_{k,j})}{1+\varepsilon|\xi_{\min}|}$ in (4.27) is zero and there is no change for the weights to build $\boldsymbol{\omega}_k(i)$. The main steps for the proposed SI–LMS and SI–RLS algorithms are listed as follows:

Step 1 Each node carries out the adaptation through its local information based on the LMS or RLS algorithm.

Step 2 Each node calculates the error pattern through (4.19).

Step 3 Each node modifies the error vector $\hat{\boldsymbol{e}}_k$.

Step 4 Each node combines the information obtained from its selected neighbors through (4.27).

The SI–LMS and SI–RLS algorithms are detailed in Table 4.3. For the ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms, we design different combination steps and employ the same adaptation procedure, which means the proposed algorithms have the ability to equip any diffusion–type wireless networks operating with other than the LMS and RLS algorithms. This includes, for example, the diffusion conjugate gradient strategy [71].

4.4 Analysis of the proposed algorithms

In this section, a statistical analysis of the proposed algorithms is developed, including a stability analysis and an MSE analysis of the steady–state and tracking performance. In addition, the computational complexity of the proposed algorithms is also detailed.

Table 4.3: The SI-LMS and SI-RLS Algorithms

Initialize: $\boldsymbol{\omega}_k(-1)$=0, $k = 1, 2, \ldots, N$

$\boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I}, \delta =$ small positive constant

For each time instant $i = 1, 2, ..., I$

For each node $k = 1, 2, \ldots, N$

The adaptation step for computing $\boldsymbol{\psi}_k(i)$

is exactly the same as the ES-LMS and ES-RLS

for the SI-LMS and SI-RLS algorithms respectively

end

For each node $k = 1, 2, \ldots, N$

$e_{kl}(i) = d_k(i) - \boldsymbol{\psi}_l^H(i)\boldsymbol{x}_k(i) \quad l \in \mathcal{N}_k$

$\boldsymbol{c}_k = [c_{kl}] \quad l \in \mathcal{N}_k$

$\boldsymbol{\Psi}_k = [\boldsymbol{\psi}_l(i)] \quad l \in \mathcal{N}_k$

$\hat{\boldsymbol{e}}_k = [e_{kl}(i)] \quad l \in \mathcal{N}_k$

Find the maximum and minimum absolute terms in $\boldsymbol{e}_k$

Modified $\hat{\boldsymbol{e}}_k$ as $\hat{\boldsymbol{e}}_k$=[$0 \cdots 0, \underbrace{|e_{kl}(i)|}_{\max}, 0 \cdots 0, \underbrace{-|e_{kl}(i)|}_{\min}, 0 \cdots 0$]

$\xi_{\min} = \min\left(|e_{kl}(i)|\right)$

$\boldsymbol{\omega}_k(i) = \sum_{j=1}^{|\mathcal{N}_k|} \left[c_{k,j} - \rho\varepsilon\frac{\text{sign}(e_{k,j})}{1+\varepsilon|\xi_{\min}|}\right]\boldsymbol{\psi}_{k,j}$

end

end

Before we start the analysis, we make some assumptions that are common in the literature [9].

Assumption I: The weight-error vector $\boldsymbol{\varepsilon}_k(i)$ and the input signal vector $\boldsymbol{x}_k(i)$ are statistically independent, and the weight–error vector for node k is defined as

$$\boldsymbol{\varepsilon}_k(i) \triangleq \boldsymbol{\omega}_k(i) - \boldsymbol{\omega}_0, \tag{4.28}$$

where $\boldsymbol{\omega}_0$ denotes the optimum Wiener solution of the actual parameter vector to be estimated, and $\boldsymbol{\omega}_k(i)$ is the estimate produced by a proposed algorithm at time instant i.

Assumption II: The input signal vector $x_l(i)$ is drawn from a stochastic process, which is ergodic in the autocorrelation function [9].

Assumption III: The $M \times 1$ vector $\boldsymbol{q}(i)$ represents a stationary sequence of independent zero–mean vectors and positive definite autocorrelation matrix $\boldsymbol{Q} = \mathbb{E}[\boldsymbol{q}(i)\boldsymbol{q}^H(i)]$, which is independent of $\boldsymbol{x}_k(i)$, $\boldsymbol{n}_k(i)$ and $\boldsymbol{\varepsilon}_l(i)$.

Assumption IV (Independence): All regressor input signals $\boldsymbol{x}_k(i)$ are spatially and temporally independent. This assumption allows us to consider the input signal $\boldsymbol{x}_k(i)$ independent of $\boldsymbol{\omega}_l(i), l \in \mathcal{N}_k$.

4.4.1 Stability Analysis

In general, to ensure that a partially-connected network performance can converge to the global network performance, the estimates should be propagated across the network [21]. The work in [64] shows that it is central to the performance that each node should be able to reach the other nodes through one or multiple hops [21].

To discuss the stability analysis of the proposed ES–LMS and SI–LMS algorithms, we

first substitute (4.6) into (4.17) and obtain

$$\begin{aligned}\boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i+1)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\omega}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\omega}_0 + \boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &= \sum_{l\in\widehat{\Omega}_k(i)} \boldsymbol{\omega}_0 c_{kl} + \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i)\\ &\text{subject to } \sum_l c_{kl}(i) = 1\\ &= \boldsymbol{\omega}_0 + \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i). \end{aligned} \tag{4.29}$$

Then, we have

$$\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\widehat{\Omega}_k(i)} [\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_l^*(i+1)]c_{kl}(i). \tag{4.30}$$

By employing *Assumption IV*, we start with (4.30) for the ES–LMS algorithm and define the global vectors and matrices:

$$\boldsymbol{\varepsilon}(i+1) \triangleq [\boldsymbol{\varepsilon}_1(i+1), \cdots, \boldsymbol{\varepsilon}_N(i+1)]^T \tag{4.31}$$

$$\boldsymbol{M} \triangleq \text{diag}\{\mu_1 \boldsymbol{I}_M, ..., \mu_N \boldsymbol{I}_M\} \tag{4.32}$$

$$\boldsymbol{D}(i+1) \triangleq \text{diag}\{\boldsymbol{x}_1(i+1)\boldsymbol{x}_1^H(i+1), ..., \boldsymbol{x}_N(i+1)\boldsymbol{x}_N^H(i+1)\} \tag{4.33}$$

and the $NM \times 1$ vector

$$\boldsymbol{g}(i+1) = [\boldsymbol{x}_1^T(i+1)n_1(i+1), \cdots, \boldsymbol{x}_N^T(i+1)n_N(i+1)]^T. \tag{4.34}$$

We also define an $N \times N$ matrix $\boldsymbol{C}$ where the combining coefficients $\{c_{kl}\}$ correspond to the $\{l, k\}$ entries of the matrix $\boldsymbol{C}$ and the $NM \times NM$ matrix $\boldsymbol{C}_G$ with a Kronecker structure:

$$\boldsymbol{C}_G = \boldsymbol{C} \otimes \boldsymbol{I}_M \tag{4.35}$$

where $\otimes$ denotes the Kronecker product.

By inserting $e_l(i+1) = e_{0-l}(i+1) - \boldsymbol{\varepsilon}_l^H(i)\boldsymbol{x}_l(i+1)$ into (4.30), the global version of (4.30) can then be written as

$$\boldsymbol{\varepsilon}(i+1) = \boldsymbol{C}_G^T\big[\boldsymbol{I} - \boldsymbol{M}\boldsymbol{D}(i+1)\big]\boldsymbol{\varepsilon}(i) + \boldsymbol{C}_G^T\boldsymbol{M}\boldsymbol{g}(i+1), \tag{4.36}$$

where $e_{0-l}(i+1)$ is the estimation error produced by the Wiener filter for node l as described by

$$e_{0-l}(i+1) = d_l(i) - \boldsymbol{\omega}_0^H \boldsymbol{x}_l(i). \tag{4.37}$$

If we define

$$\begin{aligned} \mathcal{D} &\triangleq \mathbb{E}[\boldsymbol{D}(i+1)] \\ &= \text{diag}\{\boldsymbol{R}_1, ..., \boldsymbol{R}_N\} \end{aligned} \tag{4.38}$$

and take the expectation of (4.36), we arrive at

$$\mathbb{E}\{\boldsymbol{\varepsilon}(i+1)\} = \boldsymbol{C}_G^T\big[\boldsymbol{I} - \boldsymbol{M}\mathcal{D}\big]\mathbb{E}\{\boldsymbol{\varepsilon}(i)\}. \tag{4.39}$$

Before we proceed, let us define $\boldsymbol{X} = \boldsymbol{I} - \boldsymbol{M}\mathcal{D}$. We say that a square matrix $\boldsymbol{X}$ is stable if it satisfies $\boldsymbol{X}^i \to 0$ as $i \to \infty$. A known result in linear algebra states that a matrix is stable if, and only if, all its eigenvalues lie inside the unit circle. We need the following lemma to proceed [17].

Lemma 1: Let $\boldsymbol{C}_G$ and $\boldsymbol{X}$ denote arbitrary $NM \times NM$ matrices, where $\boldsymbol{C}_G$ has real, non-negative entries, with columns adding up to one. Then, the matrix $\boldsymbol{Y} = \boldsymbol{C}_G^T\boldsymbol{X}$ is stable for any choice of $\boldsymbol{C}_G$ if, and only if, $\boldsymbol{X}$ is stable.

Proof: Assume that $\boldsymbol{X}$ is stable, it is true that for every square matrix $\boldsymbol{X}$ and every $\alpha > 0$, there exists a submultiplicative matrix norm $||\cdot||_\tau$ that satisfies $||\boldsymbol{X}||_\tau \leq \tau(\boldsymbol{X}) + \alpha$, where the submultiplicative matrix norm $||\boldsymbol{AB}|| \leq ||\boldsymbol{A}|| \cdot ||\boldsymbol{B}||$ holds and $\tau(\boldsymbol{X})$ is the spectral radius of $\boldsymbol{X}$ [10, 72]. Since $\boldsymbol{X}$ is stable, $\tau(\boldsymbol{X}) < 1$, and we can choose $\alpha > 0$ such that $\tau(\boldsymbol{X}) + \alpha = v < 1$ and $||\boldsymbol{X}||_\tau \leq v < 1$. Then we obtain [17]

$$\begin{aligned} ||\boldsymbol{Y}^i||_\tau &= ||(\boldsymbol{C}_G^T\boldsymbol{X})^i||_\tau \\ &\leq ||(\boldsymbol{C}_G^T)^i||_\tau \cdot ||\boldsymbol{X}^i||_\tau \\ &\leq v^i||(\boldsymbol{C}_G^T)^i||_\tau. \end{aligned} \tag{4.40}$$

Since $\boldsymbol{C}_G^T$ has non–negative entries with columns that add up to one, it is element–wise bounded by unity. This means its Frobenius norm is bounded as well and by the equivalence of norms, so is any norm, in particular $||(\boldsymbol{C}_G^T)^i||_\tau$. As a result, we have

$$\lim_{i\to\infty} ||\boldsymbol{Y}^i||_\tau = \boldsymbol{0}, \tag{4.41}$$

so $\boldsymbol{Y}^i$ converges to the zero matrix for large i. Therefor, $\boldsymbol{Y}$ is stable.

In view of *Lemma 1* and (82), we need the matrix $\boldsymbol{I} - \boldsymbol{M}\mathcal{D}$ to be stable. As a result, it requires $\boldsymbol{I} - \mu_k \boldsymbol{R}_k$ to be stable for all k, which holds if the following condition is satisfied:

$$0 < \mu_k < \frac{2}{\lambda_{max}\big(\boldsymbol{R}_k\big)} \tag{4.42}$$

where $\lambda_{max}\big(\boldsymbol{R}_k\big)$ is the largest eigenvalue of the correlation matrix $\boldsymbol{R}_k$. The difference between the ES–LMS and SI–LMS algorithms is the strategy to calculate the matrix $\boldsymbol{C}$. *Lemma 1* indicates that for any choice of $\boldsymbol{C}$, only $\boldsymbol{X}$ needs to be stable. As a result, SI–LMS has the same convergence condition as in (4.42). Given the convergence conditions, the proposed ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms will adapt according to the network connectivity by choosing the group of nodes with the best available performance to construct their estimates.

4.4.2 MSE Steady–State Analysis

In this part of the analysis, we devise formulas to predict the excess MSE (EMSE) of the proposed algorithms. The error signal at node k can be expressed as

$$\begin{aligned} e_k(i) &= d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i) \\ &= d_k(i) - [\boldsymbol{\omega}_0 - \boldsymbol{\varepsilon}_k(i)]^H \boldsymbol{x}_k(i) \\ &= d_k(i) - \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i) \\ &= e_{0-k} + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i). \end{aligned} \tag{4.43}$$

With *Assumption I*, the MSE expression can be derived as

$$\begin{aligned} \mathcal{J}_{mse-k}(i) &= \mathbb{E}[|e_k(i)|^2] \\ &= \mathbb{E}\Big[\big(e_{0-k} + \boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\big)\big(e_0^* + \boldsymbol{x}_k^H(i)\boldsymbol{\varepsilon}_k(i)\big)\Big] \\ &= \mathcal{J}_{min-k} + \mathbb{E}[\boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{\varepsilon}_k(i)] \\ &= \mathcal{J}_{min-k} + \text{tr}\{\mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]\} \\ &= \mathcal{J}_{min-k} + \text{tr}\{\mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]\mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)]\} \\ &= \mathcal{J}_{min-k} + \text{tr}\{\boldsymbol{R}_k(i)\boldsymbol{K}_k(i)\}, \end{aligned} \tag{4.44}$$

where tr$(\cdot)$ denotes the trace of a matrix and $\mathcal{J}_{min-k}$ is the minimum mean–square error (MMSE) for node k [9]:

$$\mathcal{J}_{min-k} = \sigma_{d,k}^2 - \boldsymbol{p}_k^H(i)\boldsymbol{R}_k^{-1}(i)\boldsymbol{p}_k(i), \tag{4.45}$$

$\boldsymbol{R}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]$ is the correlation matrix of the inputs for node k, $\boldsymbol{p}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)d_k^*(i)]$ is the cross–correlation vector between the inputs and the measurement $d_k(i)$, and $\boldsymbol{K}_k(i) = \mathbb{E}[\boldsymbol{\varepsilon}_k(i)\boldsymbol{\varepsilon}_k^H(i)]$ is the weight–error correlation matrix. From [9], the EMSE is defined as the difference between the mean–square error at time instant i and the minimum mean–square error. Then, we can write

$$\begin{aligned} \mathcal{J}_{ex-k}(i) &= \mathcal{J}_{mse-k}(i) - \mathcal{J}_{min-k} \\ &= \text{tr}\{\boldsymbol{R}_k(i)\boldsymbol{K}_k(i)\}. \end{aligned} \tag{4.46}$$

For the proposed adaptive link selection algorithms, we will derive the EMSE formulas separately based on (4.46) and *we assume that the input signal is modeled as a stationary process*.

ES–LMS

To update the estimate $\boldsymbol{\omega}_k(i)$, we employ

$$\begin{aligned} \boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i+1) \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \mu_l\boldsymbol{x}_l(i+1)e_l^*(i+1)] \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \mu_l\boldsymbol{x}_l(i+1)(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))]. \end{aligned} \tag{4.47}$$

Then, subtracting $\boldsymbol{\omega}_0$ from both sides of (4.47), we arrive at

$$\begin{aligned}
\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \mu_l \boldsymbol{x}_l(i+1)(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] \\
&\quad - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\omega}_0 \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)\big(d_l(i+1) - \boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i)+\boldsymbol{\omega}_0)\big)\Big] \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)\big(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\varepsilon}_l(i) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_0\big)\Big] \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i) - \mu_l \boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big] \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\big(\boldsymbol{I} - \mu_l \boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big].
\end{aligned} \tag{4.48}$$

Let us introduce the random variables $\alpha_{kl}(i)$:

$$\alpha_{kl}(i) = \begin{cases} 1, \text{ if } l \in \widehat{\Omega}_k(i) \\ 0, \text{ otherwise.} \end{cases} \tag{4.49}$$

At each time instant, each node will generate data associated with network covariance matrices $\boldsymbol{A}_k$ with size $N \times N$ which reflect the network topology, according to the exhaustive search strategy. In the network covariance matrices $\boldsymbol{A}_k$, a value equal to 1 means nodes k and l are linked and a value 0 means nodes k and l are not linked.

For example, suppose a network has 5 nodes. For node 3, there are two neighbor nodes, namely, nodes 2 and 5. Through Eq. (4.12), the possible configurations of set Ω_3 are $\{3,2\}, \{3,5\}$ and $\{3,2,5\}$. Evaluating all the possible sets for Ω_3, the relevant covariance matrices $\boldsymbol{A}_3$ with size 5×5 at time instant i are described in Fig. 4.3.

Then, the coefficients α_{kl} are obtained according to the covariance matrices $\boldsymbol{A}_k$. In this example, the three sets of α_{kl} are respectively shown in Table 4.4.

The parameters c_{kl} will then be calculated through Eq. (2.16) for different choices of matrices $\boldsymbol{A}_k$ and coefficients α_{kl}. After α_{kl} and c_{kl} are calculated, the error pattern for each possible Ω_k will be calculated through (4.15) and the set with the smallest error will be selected according to (4.16).

(a){3,2}

		0		
		1		
0	1	1	0	0
		0		
		0		

(a){3,5}

		0		
		0		
0	0	1	0	1
		0		
		1		

(c){3,2,5}

		0		
		1		
0	1	1	0	1
		0		
		1		

Figure 4.3: Covariance matrices $\boldsymbol{A}_3$ for different sets of Ω_3

Table 4.4: Coefficients α_{kl} for different sets of Ω_3

$$\{2,3\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=1\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=0\end{cases}\qquad \{3,5\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=0\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=1\end{cases}\qquad \{2,3,5\}\begin{cases}\alpha_{31}=0\\ \alpha_{32}=1\\ \alpha_{33}=1\\ \alpha_{34}=0\\ \alpha_{35}=1\end{cases}$$

With the newly defined α_{kl}, (4.48) can be rewritten as

$$\boldsymbol{\varepsilon}_k(i+1)=\sum_{l\in\mathcal{N}_k}\alpha_{kl}(i)c_{kl}(i)\Big[\big(\boldsymbol{I}-\mu_l\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i)+\mu_l\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big]. \tag{4.50}$$

Starting from (4.46), we then focus on $\boldsymbol{K}_k(i+1)$.

$$\boldsymbol{K}_k(i+1)=\mathbb{E}[\boldsymbol{\varepsilon}_k(i+1)\boldsymbol{\varepsilon}_k^H(i+1)]. \tag{4.51}$$

In (4.50), the term $\alpha_{kl}(i)$ is determined through the network topology for each subset, while the term $c_{kl}(i)$ is calculated through the Metropolis rule. We assume that $\alpha_{kl}(i)$ and $c_{kl}(i)$ are statistically independent from the other terms in (4.50). Upon convergence, the parameters $\alpha_{kl}(i)$ and $c_{kl}(i)$ do not vary because at steady state the choice of the subset

$\widehat{\Omega}_k(i)$ for each node k will be fixed. Then, under these assumptions, substituting (4.50) into (4.51) we arrive at:

$$\begin{aligned}\boldsymbol{K}_k(i+1) = \sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Bigg(\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\boldsymbol{K}_l(i)\times\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\\ +\mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\times\boldsymbol{R}_l(i+1)\Bigg)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Bigg(\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I}-\mu_q\boldsymbol{R}_l(i+1)\big)\\ +\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\boldsymbol{R}_{l,q}(i+1)\Bigg)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Bigg(\big(\boldsymbol{I}-\mu_q\boldsymbol{R}_q(i+1)\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{R}_l(i+1)\big)\\ +\mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\boldsymbol{R}_{l,q}^H(i+1)\Bigg)\end{aligned}\tag{4.52}$$

where $\boldsymbol{R}_{l,q}(i+1)=\mathbb{E}[\boldsymbol{x}_l(i+1)\boldsymbol{x}_q^H(i+1)]$ and $\boldsymbol{K}_{l,q}(i)=\mathbb{E}[\boldsymbol{\varepsilon}_l(i)\boldsymbol{\varepsilon}_q^H(i)]$. To further simplify the analysis, we assume that the samples of the input signal $\boldsymbol{x}_k(i)$ are uncorrelated, i.e., $\boldsymbol{R}_k=\sigma_{x,k}^2\boldsymbol{I}$ with $\sigma_{x,k}^2$ being the variance. Using the diagonal matrices $\boldsymbol{R}_k=\boldsymbol{\Lambda}_k=\sigma_{x,k}^2\boldsymbol{I}$ and $\boldsymbol{R}_{l,q}=\boldsymbol{\Lambda}_{l,q}=\sigma_{x,l}\sigma_{x,q}\boldsymbol{I}$ we can write

$$\begin{aligned}\boldsymbol{K}_k(i+1) = \sum_{l\in\mathcal{N}_k}\mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Bigg(\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\boldsymbol{K}_l(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\\ +\mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\boldsymbol{\Lambda}_l\Bigg)\\ +\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\times\Bigg(\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I}-\mu_q\boldsymbol{\Lambda}_q\big)\\ +\mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\boldsymbol{\Lambda}_{l,q}\Bigg)+\sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\\ \times\Bigg(\big(\boldsymbol{I}-\mu_q\boldsymbol{\Lambda}_q\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I}-\mu_l\boldsymbol{\Lambda}_l\big)+\mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\boldsymbol{\Lambda}_{l,q}^H\Bigg).\end{aligned}\tag{4.53}$$

Due to the structure of the above equations, the approximations and the quantities in-

volved, we can decouple (4.53) into

$$
\begin{aligned}
K_k^n(i+1) = & \sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Bigg(\big(1-\mu_l\lambda_l^n\big)K_l^n(i)\big(1-\mu_l\lambda_l^n\big) \\
& + \mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\lambda_l^n\Bigg) \\
& + \sum_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\Bigg(\big(1-\mu_l\lambda_l^n\big)K_{l,q}^n(i)\big(1-\mu_q\lambda_q^n\big) \\
& + \mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\lambda_{l,q}^n\Bigg) + \sum_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big] \\
& \times\Bigg(\big(1-\mu_q\lambda_q^n\big)(K_{l,q}^n(i))^H\big(1-\mu_l\lambda_l^n\big) + \mu_l\mu_q e_{0-q}(i+1)e_{0-l}^*(i+1)\lambda_{l,q}^n\Bigg),
\end{aligned}
\tag{4.54}
$$

where $K_k^n(i+1)$ is the nth element of the main diagonal of $\boldsymbol{K}_k(i+1)$. With the assumption that $\alpha_{kl}(i)$ and $c_{kl}(i)$ are statistically independent from the other terms in (4.50), we can rewrite (4.54) as

$$
\begin{aligned}
K_k^n(i+1) = & \sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)\Big]\mathbb{E}\Big[c_{kl}^2(i)\Big]\Bigg(\big(1-\mu_l\lambda_l^n\big)^2K_l^n(i) + \mu_l^2 e_{0-l}(i+1)e_{0-l}^*(i+1)\lambda_l^n\Bigg) \\
& + 2\times\sum_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)\Big]\mathbb{E}\Big[c_{kl}(i)c_{kq}(i)\Big]\Bigg(\big(1-\mu_l\lambda_l^n\big)\big(1-\mu_q\lambda_q^n\big)K_{l,q}^n(i) \\
& + \mu_l\mu_q e_{0-l}(i+1)e_{0-q}^*(i+1)\lambda_{l,q}^n\Bigg).
\end{aligned}
\tag{4.55}
$$

By taking $i\to\infty$, we can obtain (4.56).

$$
K_k^n(\text{ES-LMS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\mu_l^2\mathcal{J}_{min-l}\lambda_l^n + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\mu_l\mu_q e_{0-l}e_{0-q}^*\lambda_{l,q}^n}{1-\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2(1-\mu_l\lambda_l^n)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}(1-\mu_l\lambda_l^n)(1-\mu_q\lambda_q^n)}.
\tag{4.56}
$$

We assume that the choice of covariance matrix $\boldsymbol{A}_k$ for node k is fixed upon the proposed algorithms convergence, as a result, the covariance matrix $\boldsymbol{A}_k$ is deterministic and does not vary. In the above example, we assume the choice of $\boldsymbol{A}_3$ is fixed as shown in Fig. 4.4.

		0		
		1		
0	1	1	0	1
		0		
		1		

Figure 4.4: Covariance matrix $\boldsymbol{A}_3$ upon convergence

Then the coefficients α_{kl} will also be fixed and given by

$$\begin{cases} \alpha_{31} = 0 \\ \alpha_{32} = 1 \\ \alpha_{33} = 1 \\ \alpha_{34} = 0 \\ \alpha_{35} = 1 \end{cases}$$

as well as the parameters c_{kl} that are computed using the Metropolis combining rule. As a result, the coefficients α_{kl} and the coefficients c_{kl} are deterministic and can be taken out from the expectation. The MSE is then given by

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(\text{ES-LMS}). \tag{4.57}$$

SI–LMS

For the SI–LMS algorithm, we do not need to consider all possible combinations. This algorithm simply adjusts the combining coefficients for each node with its neighbors in order to select the neighbor nodes that yield the smallest MSE values. Thus, we redefine the combining coefficients through (4.27)

$$c_{kl-new} = c_{kl} - \rho\varepsilon \frac{\text{sign}(|e_{kl}|)}{1 + \varepsilon|\xi_{\min}|} \quad (l \in \mathcal{N}_k). \tag{4.58}$$

For each node k, at time instant i, after it receives the estimates from all its neighbors, it calculates the error pattern $e_{kl}(i)$ for every estimate received through Eq. (4.19) and finds the nodes with the largest and smallest errors. An error vector $\hat{\boldsymbol{e}}_k$ is then defined through (4.22), which contains all error patterns $e_{kl}(i)$ for node k.

Then a procedure which is detailed after Eq. (4.22) is carried out and modifies the error vector $\hat{\boldsymbol{e}}_k$. For example, suppose node 5 has three neighbor nodes, which are nodes

$3, 6$ and 8. The error vector $\hat{\boldsymbol{e}}_5$ has the form described by $\hat{\boldsymbol{e}}_5 = [e_{53}, e_{55}, e_{56}, e_{58}] = [0.023, 0.052, -0.0004, -0.012]$. After the modification, the error vector $\hat{\boldsymbol{e}}_5$ will be edited as $\hat{\boldsymbol{e}}_5 = [0, 0.052, -0.0004, 0]$. The quantity h_{kl} is then defined as

$$h_{kl} = \rho\varepsilon \frac{\text{sign}(|e_{kl}|)}{1 + \varepsilon|\xi_{\min}|} \ \ (l \in \mathcal{N}_k), \tag{4.59}$$

and the term 'error pattern' e_{kl} in (4.59) is from the modified error vector $\hat{\boldsymbol{e}}_k$.

From [70], we employ the relation $\mathbb{E}[\text{sign}(e_{kl})] \approx \text{sign}(e_{0-k})$. According to Eqs. (4.1) and (4.37), when the proposed algorithm converges at node k or the time instant i goes to infinity, we assume that the error e_{0-k} will be equal to the noise variance at node k. Then, the asymptotic value h_{kl} can be divided into three situations due to the rule of the SI–LMS algorithm:

$$h_{kl} = \begin{cases} \rho\varepsilon \frac{\text{sign}(|e_{0-k}|)}{1+\varepsilon|e_{0-k}|} & \text{for the node with the largest MSE} \\ \rho\varepsilon \frac{\text{sign}(-|e_{0-k}|)}{1+\varepsilon|e_{0-k}|} & \text{for the node with the smallest MSE} \\ 0 & \text{for all the remaining nodes.} \end{cases} \tag{4.60}$$

Under this situation, after the time instant i goes to infinity, the parameters h_{kl} for each neighbor node of node k can be obtained through (4.60) and the quantity h_{kl} will be deterministic and can be taken out from the expectation.

Finally, removing the random variables $\alpha_{kl}(i)$ and inserting (4.58), (4.59) into (4.56), the asymptotic values K_k^n for the SI-LMS algorithm are obtained as in (4.61).

$$K_k^n(\text{SI-LMS}) = \frac{\sum\limits_{l\in\mathcal{N}_k} (c_{kl} - h_{kl})^2 \mu_l^2 \mathcal{J}_{min-l} \lambda_l^n + 2 \sum\limits_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}} (c_{kl} - h_{kl})(c_{kq} - h_{kq}) \mu_l \mu_q e_{0-l} e_{0-q}^* \lambda_{l,q}^n}{1 - \sum\limits_{l\in\mathcal{N}_k} (c_{kl} - h_{kl})^2 (1 - \mu_l \lambda_l^n)^2 - 2 \sum\limits_{\substack{l,q\in\mathcal{N}_k \\ l\neq q}} (c_{kl} - h_{kl})(c_{kq} - h_{kq})(1 - \mu_l \lambda_l^n)(1 - \mu_q \lambda_q^n)}. \tag{4.61}$$

At this point, the theoretical results are deterministic, and the MSE for SI–LMS algorithm is given by

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(\text{SI-LMS}). \tag{4.62}$$

ES–RLS

For the proposed ES–RLS algorithm, we start from (4.10), after inserting (4.10) into (4.17), we have

$$\begin{aligned}\boldsymbol{\omega}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\psi}_l(i+1) \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \boldsymbol{k}_l(i+1)e_l^*(i+1)] \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \boldsymbol{k}_l(i+1)(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))]. \end{aligned} \tag{4.63}$$

Then, subtracting the $\boldsymbol{\omega}_0$ from both sides of (4.47), we arrive at

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \boldsymbol{k}_l(i+1)(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] \\ &\quad - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\omega}_0 \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i) + \boldsymbol{k}_l(i+1)\big(d_l(i+1) - \boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i) + \boldsymbol{\omega}_0)\big)\Big] \\ &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\big(\boldsymbol{I} - \boldsymbol{k}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) + \boldsymbol{k}_l(i+1)e_{0-l}^*(i+1)\Big]. \end{aligned} \tag{4.64}$$

Then, with the random variables $\alpha_{kl}(i)$, (4.64) can be rewritten as

$$\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\mathcal{N}_k} \alpha_{kl}(i)c_{kl}(i)\Big[\big(\boldsymbol{I} - \boldsymbol{k}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) + \boldsymbol{k}_l(i+1)e_{0-l}^*(i+1)\Big]. \tag{4.65}$$

Since $\boldsymbol{k}_l(i+1) = \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)$ [9], we can modify the (4.65) as

$$\begin{aligned}\boldsymbol{\varepsilon}_k(i+1) = \sum_{l\in\mathcal{N}_k} \alpha_{kl}(i)c_{kl}(i)\Big[&\big(\boldsymbol{I} - \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) \\ &+ \boldsymbol{\Phi}_l^{-1}(i+1)\boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big]. \end{aligned} \tag{4.66}$$

At this point, if we compare (4.66) with (4.50), we can find that the difference between (4.66) and (4.50) is, the $\boldsymbol{\Phi}_l^{-1}(i+1)$ in (4.66) has replaced the μ_l in (4.50). From [9], we also have

$$\mathbb{E}[\boldsymbol{\Phi}_l^{-1}(i+1)] = \frac{1}{i-M}\boldsymbol{R}_l^{-1}(i+1) \quad \text{for } i > M+1. \tag{4.67}$$

As a result, we can arrive

$$\boldsymbol{K}_k(i+1) = \sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Bigg(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big)\boldsymbol{K}_l(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l\boldsymbol{\Lambda}_l^{-1}}{i-M}\big) \tag{4.68}$$

$$+ \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l\boldsymbol{\Lambda}_l^{-1}}{(i-M)^2}e_{0-l}(i+1)e_{0-l}^*(i+1)\Bigg) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]$$

$$\times\Bigg(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big)\boldsymbol{K}_{l,q}(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_q\boldsymbol{\Lambda}_q^{-1}}{i-M}\big) + \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_{l,q}\boldsymbol{\Lambda}_q^{-1}}{(i-M)^2}e_{0-l}(i+1)$$

$$\times e_{0-q}^*(i+1)\Bigg) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]$$

$$\times\Bigg(\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_q\boldsymbol{\Lambda}_q^{-1}}{i-M}\big)\boldsymbol{K}_{l,q}^H(i)\big(\boldsymbol{I} - \frac{\boldsymbol{\Lambda}_l^{-1}\boldsymbol{\Lambda}_l}{i-M}\big) + \frac{\boldsymbol{\Lambda}_q^{-1}\boldsymbol{\Lambda}_{l,q}^H\boldsymbol{\Lambda}_l^{-1}}{(i-M)^2}e_{0-q}(i+1)e_{0-l}^*(i+1)\Bigg). \tag{4.69}$$

Due to the structure of the above equations, the approximations and the quantities involved, we can decouple (4.69) into

$$K_k^n(i+1) = \sum_{l\in\mathcal{N}_k} \mathbb{E}\Big[\alpha_{kl}^2(i)c_{kl}^2(i)\Big]\Bigg(\big(1 - \frac{1}{i-M}\big)^2K_l^n(i) + \frac{e_{0-l}(i+1)e_{0-l}^*(i+1)}{\lambda_l^n(i-M)^2}\Bigg)$$

$$+ \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]\Bigg(\big(1 - \frac{1}{i-M}\big)^2K_{l,q}^n(i)$$

$$+ \frac{\lambda_{l,q}^n e_{0-l}(i+1)e_{0-q}^*(i+1)}{(i-M)^2\lambda_l^n\lambda_q^n}\Bigg) + \sum_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}} \mathbb{E}\Big[\alpha_{kl}(i)\alpha_{kq}(i)c_{kl}(i)c_{kq}(i)\Big]$$

$$\times\Bigg(\big(1 - \frac{1}{i-M}\big)^2(K_{l,q}^n(i))^H + \frac{\lambda_{l,q}^n e_{0-q}(i+1)e_{0-l}^*(i+1)}{(i-M)^2\lambda_q^n\lambda_l^n}\Bigg) \tag{4.70}$$

where $K_k^n(i+1)$ is the nth elements of the main diagonals of $\boldsymbol{K}_k(i+1)$. With the assumption that, upon convergence, α_{kl} and c_{kl} do not vary, because at steady state, the choice of subset $\widehat{\Omega}_k(i)$ for each node k will be fixed, we can rewrite (4.70) as (4.71). Then, the MSE is given by

$$K_k^n(\text{ES-RLS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\dfrac{\mathcal{J}_{min-l}}{\lambda_l^n(i-M)^2} + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\dfrac{\lambda_{l,q}^n e_{0-l}e_{0-q}^*}{(i-M)^2\lambda_l^n\lambda_q^n}}{1 - \sum\limits_{l\in\mathcal{N}_k}\alpha_{kl}^2c_{kl}^2\Big(1-\dfrac{1}{i-M}\Big)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}\alpha_{kl}\alpha_{kq}c_{kl}c_{kq}\Big(1-\dfrac{1}{i-M}\Big)^2}. \tag{4.71}$$

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M}K_k^n(\text{ES-RLS}). \tag{4.72}$$

On the basis of (4.71), we have that when i tends to infinity, the MSE approaches the MMSE in theory [9].

SI–RLS

For the proposed SI–RLS algorithm, we insert (4.58) into (4.71), remove the random variables $\alpha_{kl}(i)$, and following the same procedure as for the SI–LMS algorithm, we can obtain (4.73), where h_{kl} and h_{kq} satisfy the rule in (4.60). Then, the MSE is given by

$$K_k^n(\text{SI-RLS}) = \frac{\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2\frac{\mathcal{J}_{min-l}}{\lambda_l^n(i-M)^2} + 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})\frac{\lambda_{l,q}^n e_{0-l}e_{0-q}^*}{(i-M)^2\lambda_l^n\lambda_q^n}}{1-\sum\limits_{l\in\mathcal{N}_k}(c_{kl}-h_{kl})^2\Big(1-\frac{1}{i-M}\Big)^2 - 2\sum\limits_{\substack{l,q\in\mathcal{N}_k\\ l\neq q}}(c_{kl}-h_{kl})(c_{kq}-h_{kq})\Big(1-\frac{1}{i-M}\Big)^2}. \tag{4.73}$$

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2\sum_{n=1}^{M}K_k^n(\text{SI-RLS}). \tag{4.74}$$

In conclusion, according to (4.61) and (4.73), with the help of modified combining coefficients, for the proposed SI–type algorithms, the neighbor node with lowest MSE contributes the most to the combination, while the neighbor node with the highest MSE contributes the least. Therefore, the proposed SI–type algorithms perform better than the standard diffusion algorithms with fixed combining coefficients.

4.4.3 Tracking Analysis

In this subsection, we assess the proposed ES–LMS/RLS and SI–LMS/RLS algorithms in a non–stationary environment, in which the algorithms have to track the minimum point of the error–performance surface [73, 74]. In the time–varying scenarios of interest, the optimum estimate is assumed to vary according to the model $\boldsymbol{\omega}_0(i+1) = \beta\boldsymbol{\omega}_0(i) + \boldsymbol{q}(i)$, where $\boldsymbol{q}(i)$ denotes a random perturbation [10] and $\beta = 1$ in order to facilitate the analysis. This is typical in the context of tracking analysis of adaptive algorithms [9, 10, 75, 76].

ES–LMS

For the tracking analysis of the ES–LMS algorithm, we employ *Assumption III* and start from (4.47). After subtracting the $\boldsymbol{\omega}_0(i+1)$ from both sides of (4.47), we obtain

$$\begin{aligned}
\boldsymbol{\varepsilon}_k(i+1) &= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \mu_l \boldsymbol{x}_l(i+1)(d_l(i+1) - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] \\
&\quad - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\boldsymbol{\omega}_0(i+1) \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)[\boldsymbol{\omega}_l(i) + \mu_l \boldsymbol{x}_l(i+1)(d_l(i+1) \\
&\qquad - \boldsymbol{x}_l^H(i+1)\boldsymbol{\omega}_l(i))] - \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big(\boldsymbol{\omega}_0(i) + \boldsymbol{q}(i)\Big) \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)\big(d_l(i+1) - \boldsymbol{x}_l^H(i+1)(\boldsymbol{\varepsilon}_l(i) + \boldsymbol{\omega}_0)\big)\Big] - \boldsymbol{q}(i) \\
&= \sum_{l\in\widehat{\Omega}_k(i)} c_{kl}(i)\Big[\big(\boldsymbol{I} - \mu_l \boldsymbol{x}_l(i+1)\boldsymbol{x}_l^H(i+1)\big)\boldsymbol{\varepsilon}_l(i) + \mu_l \boldsymbol{x}_l(i+1)e_{0-l}^*(i+1)\Big] - \boldsymbol{q}(i).
\end{aligned} \tag{4.75}$$

Using *Assumption III*, we can arrive at

$$\mathcal{J}_{ex-k}(i+1) = \text{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{K}_k(i+1)\} + \text{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{Q}\}. \tag{4.76}$$

The first part on the right side of (4.76), has already been obtained in the MSE steady–state analysis part in Section IV B. The second part can be decomposed as

$$\begin{aligned}
\text{tr}\{\boldsymbol{R}_k(i+1)\boldsymbol{Q}\} &= \text{tr}\Big\{\mathbb{E}\big[\boldsymbol{x}_k(i+1)\boldsymbol{x}_k^H(i+1)\big]\mathbb{E}\big[\boldsymbol{q}(i)\boldsymbol{q}^H(i)\big]\Big\} \\
&= M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}.
\end{aligned} \tag{4.77}$$

The MSE is then obtained as

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(\text{ES-LMS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{4.78}$$

SI–LMS

For the SI–LMS recursions, we follow the same procedure as for the ES-LMS algorithm, and obtain

$$\mathcal{J}_{mse-k} = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(\text{SI-LMS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{4.79}$$

ES–RLS

For the SI–RLS algorithm, we follow the same procedure as for the ES–LMS algorithm and arrive at

$$\mathcal{J}_{mse-k}(i+1) = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(i+1)(\text{ES-RLS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{4.80}$$

SI–RLS

We start from (4.74), and after a similar procedure to that of the SI–LMS algorithm, we have

$$\mathcal{J}_{mse-k}(i+1) = \mathcal{J}_{min-k} + M\sigma_{x,k}^2 \sum_{n=1}^{M} K_k^n(i+1)(\text{SI-RLS}) + M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}. \tag{4.81}$$

In conclusion, for time-varying scenarios there is only one additional term $M\sigma_{x,k}^2 \text{tr}\{\boldsymbol{Q}\}$ in the MSE expression for all algorithms, and this additional term has the same value for all algorithms. As a result, the proposed SI–type algorithms still perform better than the standard diffusion algorithms with fixed combining coefficients, according to the conclusion obtained in the previous subsection.

4.4.4 Computational Complexity

In the analysis of the computational cost of the algorithms studied, we assume complex-valued data and first analyze the adaptation step. For both ES–LMS/RLS and SI–LMS/RLS algorithms, the adaptation cost depends on the type of recursions (LMS or

Table 4.5: Computational complexity for the adaptation step per node per time instant

Adaptation Method	Multiplications	Additions
LMS	$2M+1$	$2M$
RLS	$4M^2+16M+2$	$4M^2+12M-1$

Table 4.6: Computational complexity for combination step per node per time instant

Algorithms	Multiplications	Additions
ES–LMS/RLS	$M(t+1)\frac{T!}{t!(T-t)!}$	$Mt\frac{T!}{t!(T-t)!}$
SI–LMS/RLS	$(2M+4)\lvert\mathcal{N}_k\rvert$	$(M+2)\lvert\mathcal{N}_k\rvert$

RLS) that each strategy employs. The details are shown in Table 4.5. For the combination step, we analyze the computational complexity in Table 4.6. The overall complexity for each algorithm is summarized in Table 4.7. In the above three tables, T is the total number of nodes linked to node k including node k itself and t is the number of nodes chosen from T. M is the length of the unknown vector $\boldsymbol{\omega}_0$. The proposed algorithms require extra computations as compared to the existing distributed LMS and RLS algorithms. This extra cost ranges from a small additional number of operations for the SI-LMS/RLS algorithms to a more significant extra cost that depends on T for the ES-LMS/RLS algorithms.

Table 4.7: Computational complexity per node per time instant

Algorithm	Multiplications	Additions
ES–LMS	$\left[\frac{(t+1)T!}{t!(T-t)!}+8\right]M+2$	$\left[\frac{T!}{(t-1)!(T-t)!}+8\right]M$
ES–RLS	$4M^2+\left[\frac{(t+1)T!}{t!(T-t)!}+16\right]M+2$	$4M^2+\left[\frac{T!}{(t-1)!(T-t)!}+12\right]M-1$
SI–LMS	$(8+2\mathcal{N}_k)M+4\lvert\mathcal{N}_k\rvert+2$	$(8+\lvert\mathcal{N}_k\rvert)M+2\lvert\mathcal{N}_k\rvert$
SI–RLS	$4M^2+(16+2\lvert\mathcal{N}_k\rvert)M+4\lvert\mathcal{N}_k\rvert+2$	$4M^2+(12+\lvert\mathcal{N}_k\rvert)M+2\lvert\mathcal{N}_k\rvert-1$

4.5 Simulations

In this section, we investigate the performance of the proposed link selection strategies for distributed estimation in two scenarios: wireless sensor networks and smart grids. In these applications, we simulate the proposed link selection strategies in both static and

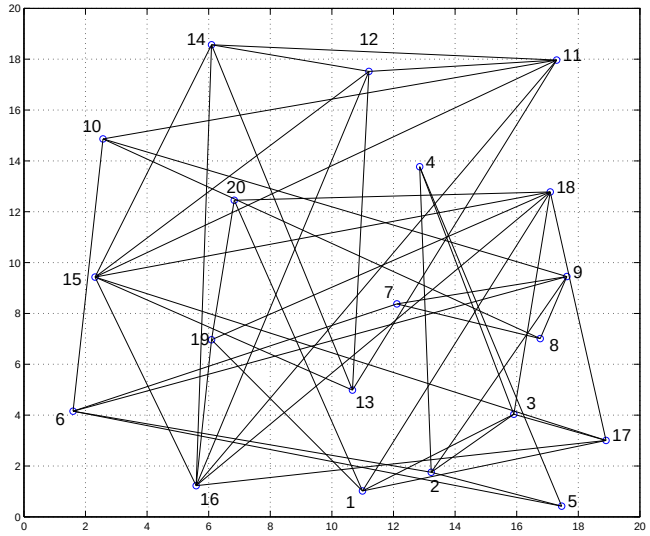

Figure 4.5: Diffusion wireless sensor networks topology with 20 nodes

time–varying scenarios. We also show the analytical results for the MSE steady–state and tracking performances that we obtained in Section 4.4.

4.5.1 Diffusion Wireless Sensor Networks

In this subsection, we compare the proposed ES–LMS/ES–RLS and SI–LMS/SI–RLS algorithms with the diffusion LMS algorithm [2], the diffusion RLS algorithm [20] and the single–link strategy [28] in terms of their MSE performance. The network topology is illustrated in Fig. 4.5 and we employ $N = 20$ nodes in the simulations. The length of the unknown parameter vector $\boldsymbol{\omega}_0$ is $M = 10$ and it is generated as a complex random vector. The input signal is generated as $\boldsymbol{x}_k(i) = [x_k(i) \quad x_k(i-1) \quad ... \quad x_k(i-M+1)]$ and $x_k(i) = u_k(i) + \alpha_k x_k(i-1)$, where α_k is a correlation coefficient and $u_k(i)$ is a white noise process with variance $\sigma_{u,k}^2 = 1 - |\alpha_k|^2$, to ensure the variance of $\boldsymbol{x}_k(i)$ is $\sigma_{x,k}^2 = 1$. The $\boldsymbol{x}_k(0)$ is defined as a Gaussian randan number with zero mean and variance $\sigma_{x,k}^2$. The noise samples are modeled as circular Gaussian noise with zero mean and variance $\sigma_{n,k}^2 \in [0.001, 0.01]$. The step size for the diffusion LMS ES–LMS and SI–LMS algorithms is $\mu = 0.2$. For the diffusion RLS algorithm, both ES–RLS and SI–RLS, the forgetting factor λ is set to 0.97 and δ is equal to 0.81. In the static scenario, the sparsity

parameters of the SI–LMS/SI–RLS algorithms are set to $\rho = 4 \times 10^{-3}$ and $\varepsilon = 10$. The Metropolis rule is used to calculate the combining coefficients c_{kl}. The MSE and MMSE are defined as in (4.3) and (4.45), respectively. The results are averaged over 100 independent runs.

In Fig. 4.6, we can see that ES–RLS has the best performance for both steady-state MSE and convergence rate, and obtains a gain of about 8 dB over the standard diffusion RLS algorithm. SI–RLS is worse than the ES–RLS, but is still significantly better than the standard diffusion RLS algorithm by about 5 dB. Regarding the complexity and processing time, SI–RLS is as simple as the standard diffusion RLS algorithm, while ES–RLS is more complex. The proposed ES–LMS and SI–LMS algorithms are superior to the standard diffusion LMS algorithm.

In the time–varying scenario, the sparsity parameters of the SI–LMS and SI–RLS algorithms are set to $\rho = 6 \times 10^{-3}$ and $\varepsilon = 10$. The unknown parameter vector $\boldsymbol{\omega}_0$ varies according to the first–order Markov vector process:

$$\boldsymbol{\omega}_0(i+1) = \beta \boldsymbol{\omega}_0(i) + \boldsymbol{q}(i), \tag{4.82}$$

where $\boldsymbol{q}(i)$ is an independent zero–mean Gaussian vector process with variance $\sigma_q^2 = 0.01$ and $\beta = 0.9998$.

Fig. 4.7 shows that, similarly to the static scenario, ES–RLS has the best performance, and obtains a 5 dB gain over the standard diffusion RLS algorithm. SI–RLS is slightly worse than the ES–RLS, but is still better than the standard diffusion RLS algorithm by about 3 dB. The proposed ES–LMS and SI–LMS algorithms have the same advantage over the standard diffusion LMS algorithm in the time-varying scenario. Notice that in the scenario with large $|\mathcal{N}_k|$, the proposed SI-type algorithms still have a better performance when compared with the standard techniques.

To illustrate the link selection for the ES–type algorithms, we provide Figs. 4.8 and 4.9. From these two figures, we can see that upon convergence the proposed algorithms converge to a fixed selected set of links $\widehat{\Omega}_k$.

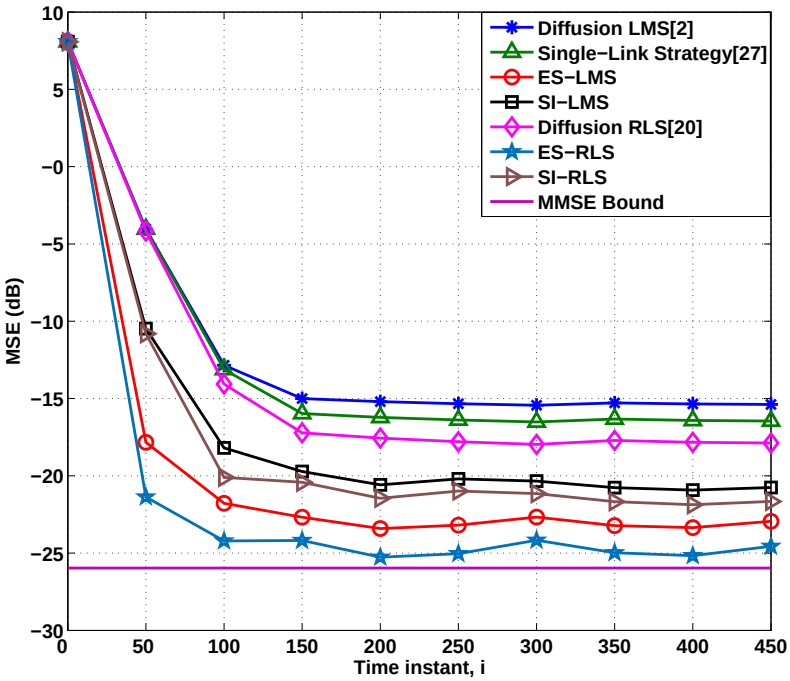

Figure 4.6: Network MSE curves in a static scenario

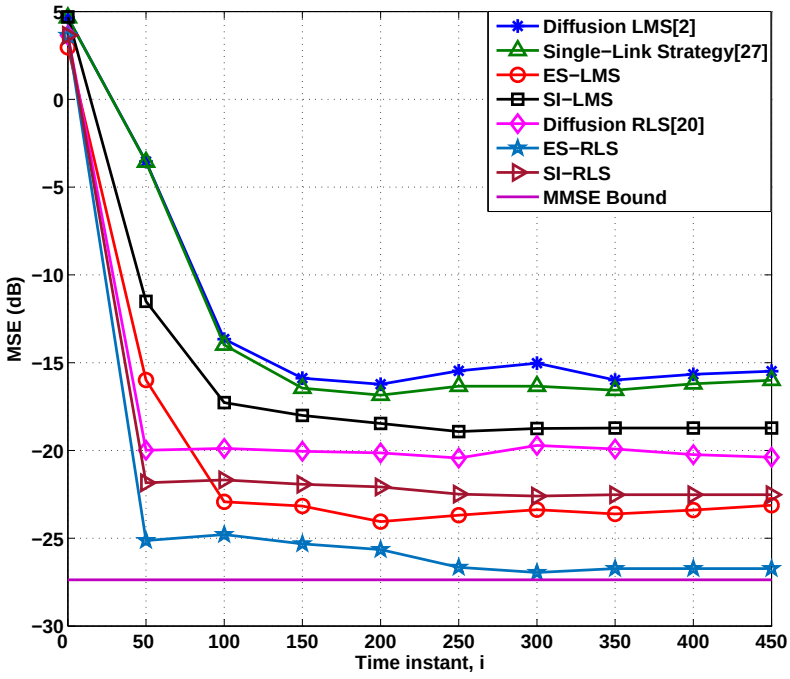

Figure 4.7: Network MSE curves in a time-varying scenario

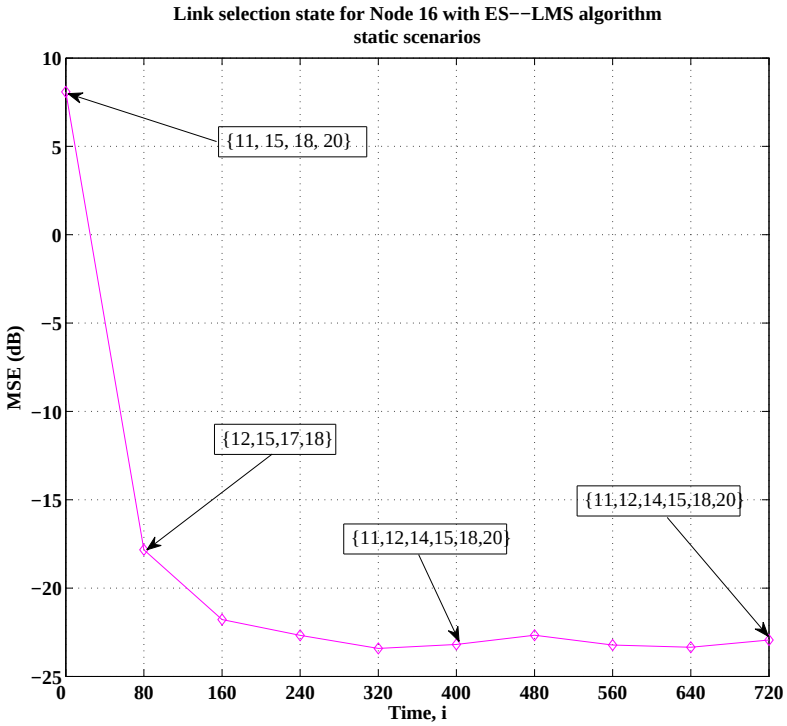

Figure 4.8: Set of selected links for node 16 with ES–LMS in a static scenario

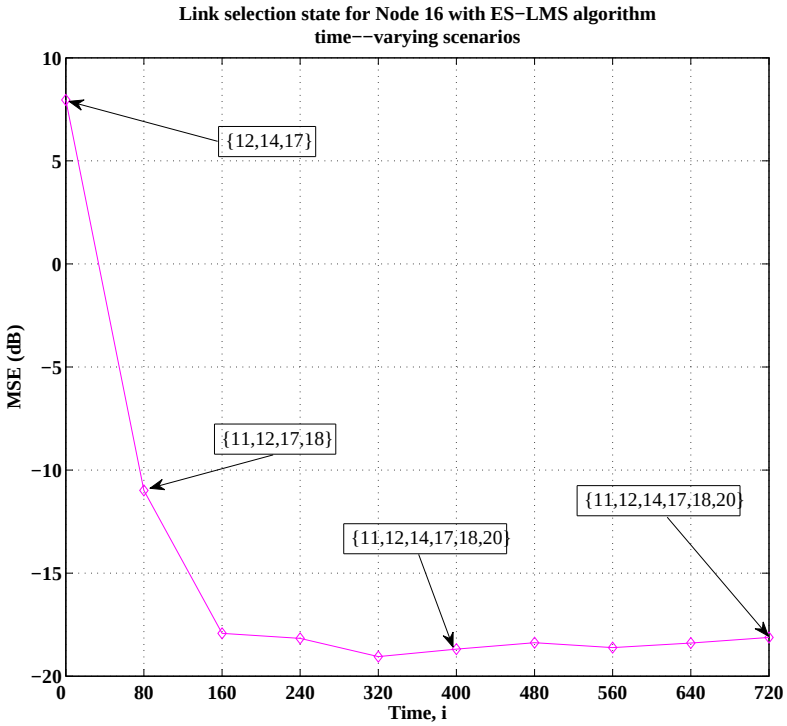

Figure 4.9: Link selection state for node 16 with ES–LMS in a time-varying scenario

4.5.2 MSE Analytical Results

The aim of this section is to validate the analytical results obtained in Section IV. First, we verify the MSE steady–state performance. Specifically, we compare the analytical results in (4.57), (4.62), (4.72) and (4.74) to the results obtained by simulations under different SNR values. The SNR indicates the input signal variance to noise variance ratio. We assess the MSE against the SNR, as show in Figs. 4.10 and 4.11. For ES–RLS and SI–RLS algorithms, we use (4.72) and (4.74) to compute the MSE after convergence. The results show that the analytical curves coincide with those obtained by simulations, which indicates the validity of the analysis. We have assessed the proposed algorithms with SNR equal to 0dB, 10dB, 20dB and 30dB, respectively, with 20 nodes in the network. For the other parameters, we follow the same definitions used to obtain the network MSE curves in a static scenario. The details have been shown on the top of each sub figure in Figs. 4.10 and 4.11. The theoretical curves for ES–LMS/RLS and SI–LMS/RLS are all close to the simulation curves.

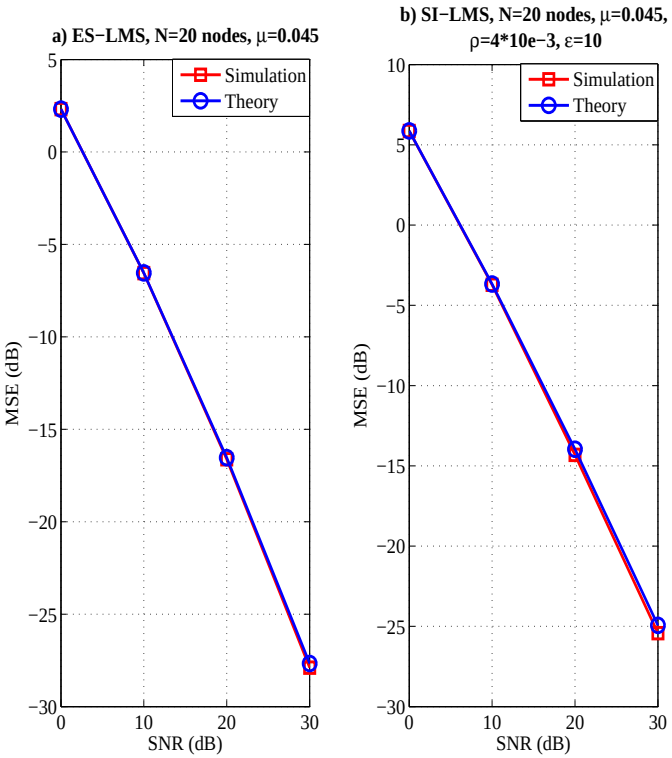

Figure 4.10: MSE steady–state performance against SNR for ES–LMS and SI–LMS

The tracking analysis of the proposed algorithms in a time–varying scenario is dis-

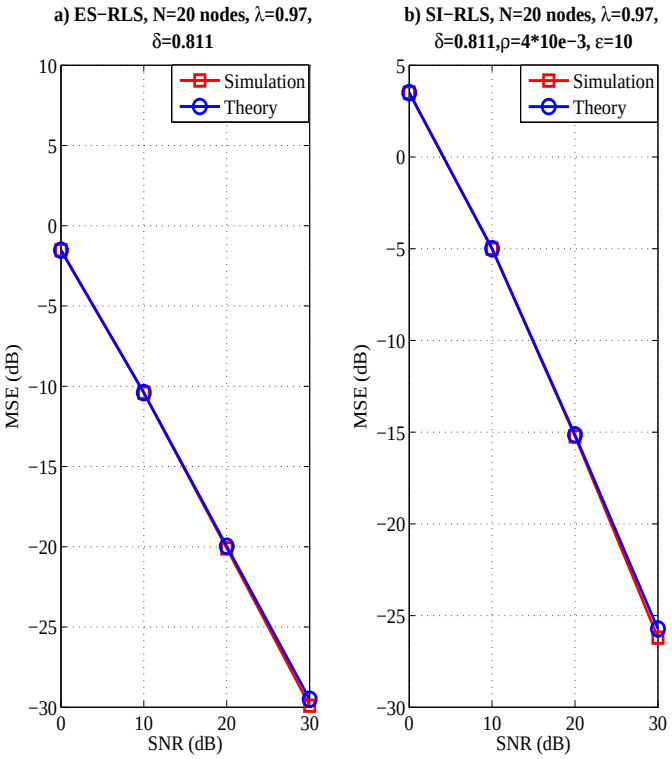

Figure 4.11: MSE steady–state performance against SNR for ES–RLS and SI–RLS

cussed as follows. Here, we verify that the results (4.78), (4.79), (4.80) and (4.81) of the subsection on the tracking analysis can provide a means of estimating the MSE. We consider the same model as in (4.82), but with β is set to 1. In the next examples, we employ $N = 20$ nodes in the network and the same parameters used to obtain the network MSE curves in a time–varying scenario. A comparison of the curves obtained by simulations and by the analytical formulas is shown in Figs. 4.12 and 4.13. From these curves, we can verify that the gap between the simulation and analysis results are extraordinary small under different SNR values. The details of the parameters are shown on the top of each sub figure in Figs. 4.12 and 4.13.

4.5.3 Smart Grids

The proposed algorithms provide a cost–effective tool that could be used for distributed state estimation in smart grid applications. In order to test the proposed algorithms in a possible smart grid scenario, we consider the IEEE 14–bus system [27], where 14 is the

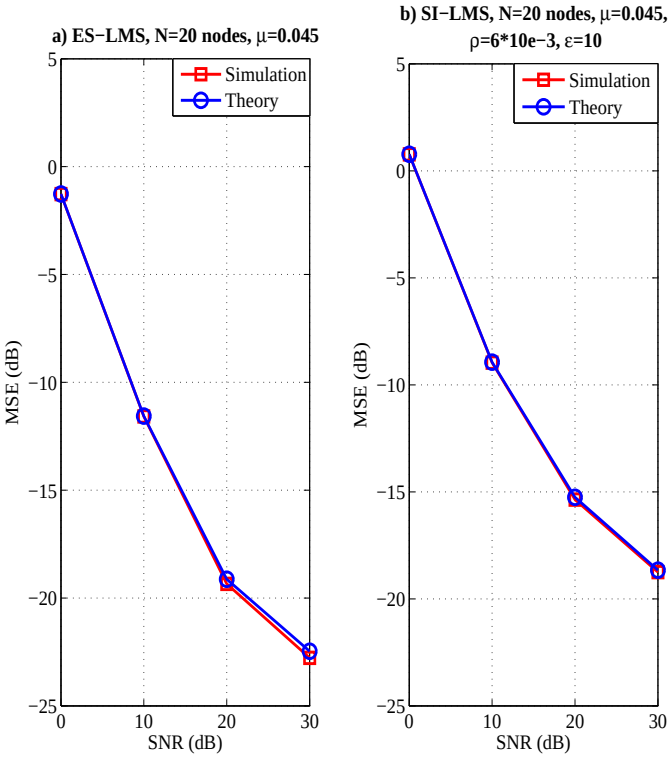

Figure 4.12: MSE steady–state performance against SNR for ES–LMS and SI–LMS in a time-varying scenario

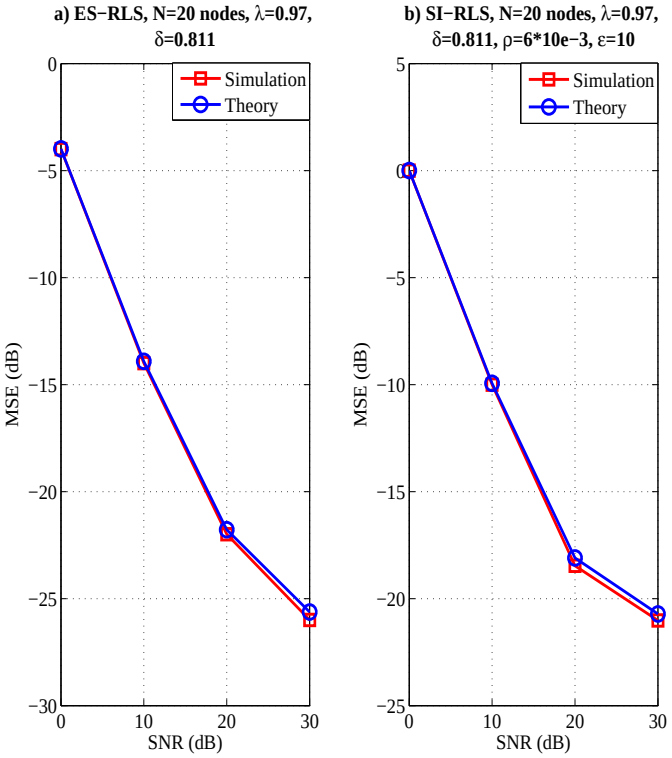

Figure 4.13: MSE steady–state performance against SNR for ES–RLS and SI–RLS in a time-varying scenario

number of substations. At every time instant i, each bus $k, k = 1, 2, \ldots, 14$, takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = X_k\big(\boldsymbol{\omega}_0(i)\big) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{4.83}$$

where $\boldsymbol{\omega}_0(i)$ is the state vector of the entire interconnected system, and $X_k(\boldsymbol{\omega}_0(i))$ is a nonlinear measurement function of bus k. The quantity $n_k(i)$ is the measurement error with mean equal to zero and which corresponds to bus k.

Initially, we focus on the linearized DC state estimation problem. The state vector $\boldsymbol{\omega}_0(i)$ is taken as the voltage phase angle vector $\boldsymbol{\omega}_0$ for all buses. Therefore, the nonlinear measurement model for state estimation (4.83) is approximated by

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{4.84}$$

where $\boldsymbol{x}_k(i)$ is the measurement Jacobian vector for bus k. Then, the aim of the distributed estimation algorithm is to compute an estimate of $\boldsymbol{\omega}_0$, which can minimize the cost function given by

$$J_k(\boldsymbol{\omega}_k(i)) = \mathbb{E}|d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)|^2. \tag{4.85}$$

We compare the proposed algorithms with the M–CSE algorithm [4], the single link strategy [28], the standard diffusion RLS algorithm [20] and the standard diffusion LMS algorithm [2] in terms of MSE performance. The MSE comparison is used to determine the accuracy of the algorithms, and compare the rate of convergence. We define the IEEE–14 bus system as in Fig. 4.14.

All buses are corrupted by additive white Gaussian noise with variance $\sigma_{n,k}^2 \in [0.001, 0.01]$. The step size for the standard diffusion LMS [2], the proposed ES–LMS and SI–LMS algorithms is 0.15. The parameter vector $\boldsymbol{\omega}_0$ is set to an all–one vector. For the diffusion RLS, ES–RLS and SI–RLS algorithms the forgetting factor λ is set to 0.945 and δ is equal to 0.001. The sparsity parameters of the SI–LMS/RLS algorithms are set to $\rho = 0.07$ and $\varepsilon = 10$. The results are averaged over 100 independent runs. We simulate the proposed algorithms for smart grids under a static scenario.

From Fig. 4.15, it can be seen that ES–RLS has the best performance, and significantly outperforms the standard diffusion LMS [2] and the $\mathcal{M}$–$\mathcal{CSE}$ [4] algorithms. The ES–

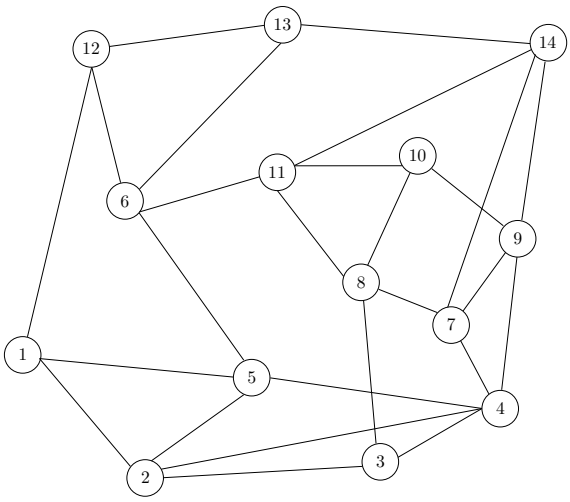

Figure 4.14: IEEE 14–bus system for simulation

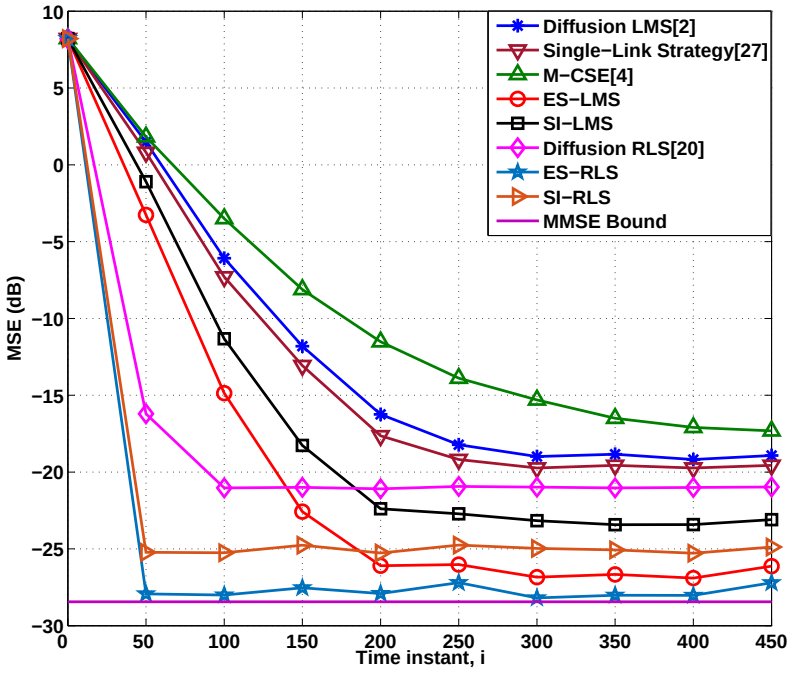

Figure 4.15: MSE performance curves for smart grids

LMS is slightly worse than ES–RLS, which outperforms the remaining techniques. SI–RLS is worse than ES–LMS but is still better than SI–LMS, while SI–LMS remains better than the diffusion RLS, LMS, $\mathcal{M}$–$\mathcal{CSE}$ algorithms and the single link strategy.

4.6 Summary

In this chapter, we have proposed ES–LMS/RLS and SI–LMS/RLS algorithms for distributed estimation in applications such as wireless sensor networks and smart grids. The proposed algorithms are based on the ideas of exhaustive search and sparsity-inspired penalty function. We have compared the proposed algorithms with existing methods. We have also devised analytical expressions to predict their MSE steady–state performance and tracking behavior. Simulation experiments have been conducted to verify the analytical results and illustrate that the proposed algorithms significantly outperform the existing strategies, in both static and time–varying scenarios, in examples of wireless sensor networks and smart grids.

Chapter 5

Distributed Compressed Estimation Based on Compressive Sensing

Contents

5.1 Introduction 101

5.2 System Model and Problem Statement 103

5.3 Proposed Distributed Compressed Estimation Scheme 104

5.4 Measurement Matrix Optimization 108

5.5 Simulations 110

5.6 Summary 113

5.1 Introduction

Distributed signal processing algorithms are of great importance for statistical inference in wireless networks and applications such as wireless sensor networks (WSNs) [2,17,47, 71]. Distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area [2]. In this context, for each node a set of neighbor nodes collect and process their local information, and transmit their estimates to a specific node. Then, each specific node combines the collected information together with its local estimate to generate improved estimates.

In many scenarios, the unknown parameter vector to be estimated can be sparse and contain only a few nonzero coefficients. Many algorithms have been developed in the literature for sparse signal estimation [3, 25, 44, 77–81]. However, these techniques are designed to take into account the full dimension of the observed data, which increases the computational cost, slows down the convergence rate and degrades mean square error (MSE) performance.

Compressive sensing (CS) [11, 12] has recently received considerable attention and been successfully applied to diverse fields, e.g., image processing [82], wireless communications [83] and MIMO radar [84]. The theory of CS states that an S–sparse parameter vector $\boldsymbol{\omega}_0$ of length M can be recovered exactly with high probability from $\mathcal{O}(S \log M)$ measurements. Mathematically, the vector $\bar{\boldsymbol{\omega}}_0$ with dimension $D \times 1$ that carries sufficient information about $\boldsymbol{\omega}_0$ ($S \leq D \ll M$) can be obtained via a linear model [12]

$$\bar{\boldsymbol{\omega}}_0 = \boldsymbol{\Gamma} \boldsymbol{\omega}_0 \tag{5.1}$$

where $\boldsymbol{\Gamma} \in R^{D \times M}$ is the measurement matrix.

The application of CS to WSNs has been recently investigated in [83, 85] and [86, 87]. A compressive wireless sensing scheme was developed in [83] to save energy and bandwidth, where CS is only employed in the transmit layer. In [85], a greedy algorithm called precognition matching pursuit was developed for CS and used at sensors and the fusion center to achieve fast reconstruction. However, the sensors are assumed to capture the target signal perfectly with only measurement noise. The work of [86] introduced a theory for distributed CS based on jointly sparse signal recovery. However, in [86] CS techniques are only applied to the transmit scenario, whereas distributed CS in the estimation scenario has not been widely investigated. A sparse model that allows the use of CS for the online recovery of large data sets in WSNs was proposed in [87], but it assumes that the sensor measurements could be gathered directly, without an estimation procedure. In summary, prior work has focused on signal reconstruction algorithms in a distributed manner but has not considered both compressed transmit strategies and estimation techniques.

In this chapter, we focus on the design of an approach that exploits lower dimensions, reduces the required bandwidth, and improves the convergence rate and the MSE performance. Inspired by CS, we introduce a scheme that incorporates compression and decompression modules into the distributed estimation procedure. In the compression module,

we compress the unknown parameter $\boldsymbol{\omega}_0$ into a lower dimension. As a result, the estimation procedure is performed in a compressed dimension. After the estimation procedure is completed, the decompression module recovers the compressed estimate into its original dimension using an orthogonal matching pursuit (OMP) algorithm [38, 43, 88]. We also present a design procedure and develop an algorithm to optimize the measurement matrices, which can further improve the performance of the proposed scheme. Specifically, we derive an adaptive stochastic gradient recursion to update the measurement matrix. Simulation results illustrate the performance of the proposed scheme and algorithm against existing techniques. In real applications, for example, in a cognitive network, distributed compressed estimation could be used to sense a sparse spectrum from the primary user, before allocating a radio resources to the seconder users [5].

5.2 System Model and Problem Statement

A wireless sensor network (WSN) with N nodes, which have limited processing capabilities, is considered with a partially connected topology. A diffusion protocol is employed although other strategies, such as incremental [1] and consensus [4] could also be used. A partially connected network means that nodes can exchange information only with their neighbors as determined by the connectivity topology. In contrast, a fully connected network means that, data broadcast by a node can be captured by all other nodes in the network [19]. At every time instant i, the sensor at each node k takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, \mathrm{I}, \tag{5.2}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ input signal vector with zero mean and variance $\sigma_{x,k}^2$, $n_k(i)$ is the noise at each node with zero mean and variance $\sigma_{n,k}^2$. From (5.2), we can see that the measurements for all nodes are related to an unknown parameter vector $\boldsymbol{\omega}_0$ with size $M \times 1$ that should be estimated by the network. We assume that $\boldsymbol{\omega}_0$ is a sparse vector with $S \ll M$ non-zero coefficients. The aim of such a network is to compute an estimate of $\boldsymbol{\omega}_0$ in a distributed fashion, which minimizes the cost function

$$J_\omega(\boldsymbol{\omega}) = \sum_{k=1}^{N} \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}^H \boldsymbol{x}_k(i)\big|^2, \tag{5.3}$$

where $\mathbb{E}$ denotes expectation. Distributed estimation of $\boldsymbol{\omega}_0$ is appealing because it provides robustness against noisy measurements and improved performance as reported in [1, 2, 4]. To solve this problem, a cost-effective technique is the adapt–then–combine (ATC) diffusion strategy [2]

$$\begin{cases} \boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i) + \mu_k \boldsymbol{x}_k(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i)\boldsymbol{x}_k(i)\big]^*, \\ \\ \boldsymbol{\omega}_k(i+1) = \sum\limits_{l \in \mathcal{N}_k} c_{kl}\boldsymbol{\psi}_l(i), \end{cases} \tag{5.4}$$

where $\mathcal{N}_k$ indicates the set of neighbors for node k, $\boldsymbol{\psi}_k(i)$ is the updated local estimator of node k, $|\mathcal{N}_k|$ denotes the cardinality of $\mathcal{N}_k$ and c_{kl} is the combination coefficient, which is calculated with respect to the Metropolis rule

$$c_{kl} = \begin{cases} \frac{1}{\max(|\mathcal{N}_k|, |\mathcal{N}_l|)}, & \text{if } k \neq l \text{ are linked} \\ 0, & \text{for } k \text{ and } l \text{ not linked} \\ 1 - \sum\limits_{l \in \mathcal{N}_k/k} c_{kl}, & \text{for } k = l \end{cases} \tag{5.5}$$

and satisfies

$$\sum_{l} c_{kl} = 1, l \in \mathcal{N}_k \forall k. \tag{5.6}$$

Existing distributed sparsity–aware estimation strategies, e.g., [3, 63, 77], are designed using the full dimension signal space, which reduces the convergence rate and degrades the MSE performance. In order to improve performance and reduce the required bandwidth, we incorporate at each node of the WSN the proposed distributed compressed estimation scheme based on CS techniques, together with a measurement matrix optimization algorithm.

5.3 Proposed Distributed Compressed Estimation Scheme

In this section, we detail the proposed distributed compressed estimation (DCE) scheme based on CS, which is depicted in Fig. 5.1. In the proposed scheme, at each node, the sensor first observes the $D \times 1$ vector $\bar{\boldsymbol{x}}_k(i)$, then with the help of the $D \times M$ measurement matrix $\boldsymbol{\Gamma}$ performs the estimation of $\boldsymbol{\omega}_0$ in the compressed domain. In other words, the

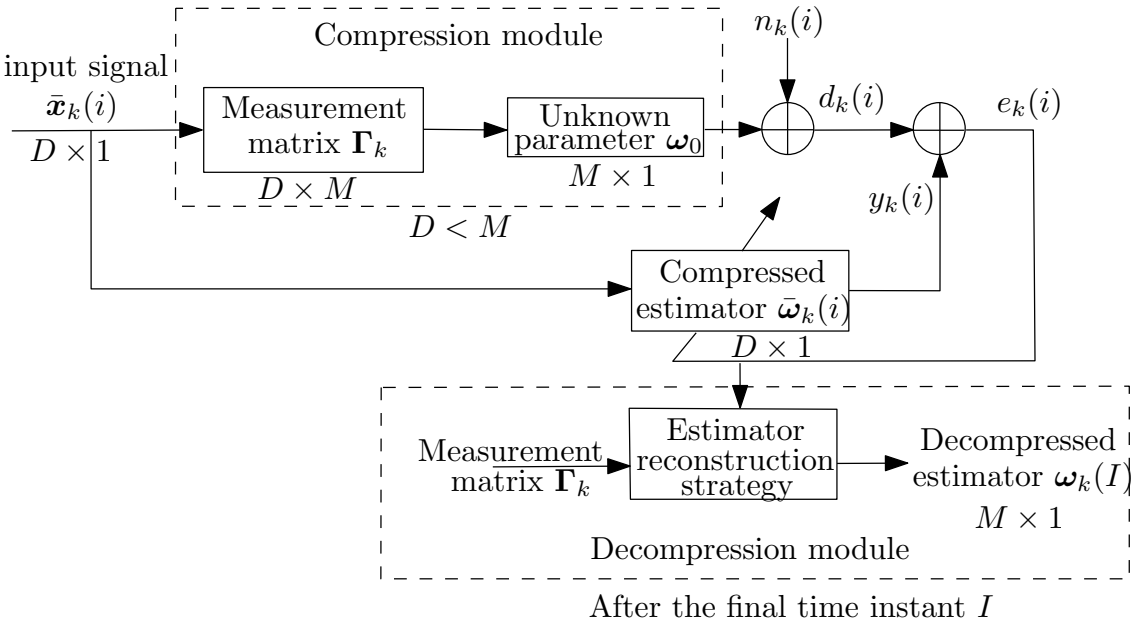

Figure 5.1: Proposed Compressive Sensing Modules

proposed scheme estimates the $D \times 1$ vector $\bar{\boldsymbol{\omega}}_0$ instead of the $M \times 1$ vector $\boldsymbol{\omega}_0$, where $D \ll M$ and the D–dimensional quantities are designated with an overbar. At each node, a decompression module employs a $D \times M$ measurement matrix $\boldsymbol{\Gamma}_k$ and a reconstruction algorithm to compute an estimate of $\boldsymbol{\omega}_0$. One advantage for the DCE scheme is that fewer parameters need to be transmitted between neighbour nodes.

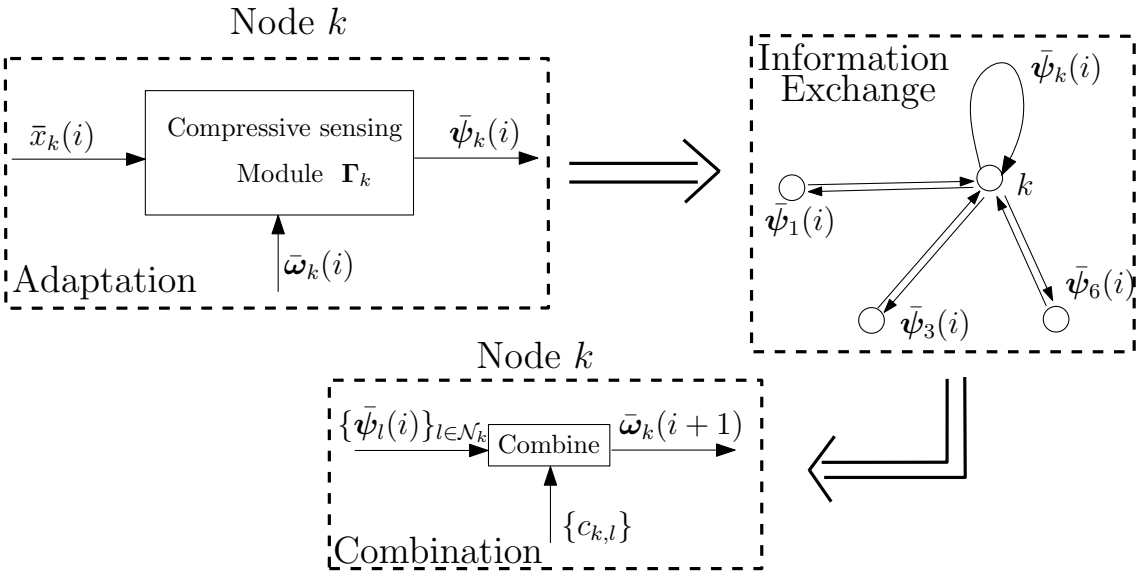

Figure 5.2: Proposed DCE Scheme

We start the description of the proposed DCE scheme with the scalar measurement

$d_k(i)$ given by

$$d_k(i) = \bar{\boldsymbol{\omega}}_0^H \bar{\boldsymbol{x}}_k(i) + n_k(i), \quad i = 1, 2, \ldots, I, \tag{5.7}$$

where $\bar{\boldsymbol{\omega}}_0 = \boldsymbol{\Gamma}_k \boldsymbol{\omega}_0$ and $\bar{\boldsymbol{x}}_k(i)$ is the $D \times 1$ input signal vector. This operation is depicted in Fig. 5.1 as the compression module.

Fig. 5.2 illustrates the proposed DCE scheme. The scheme can be divided into three steps:

1) Adaptation

In the adaptation step, at each time instant $i = 1, 2, ..., I$, each node $k = 1, 2, \ldots, N$, generates a local compressed estimator $\bar{\boldsymbol{\psi}}_k(i)$ through

$$\bar{\boldsymbol{\psi}}_k(i) = \bar{\boldsymbol{\omega}}_k(i) + \mu_k(i) e_k^*(i) \bar{\boldsymbol{x}}_k(i), \tag{5.8}$$

where $e_k(i) = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i) \bar{\boldsymbol{x}}_k(i)$ and $\mu_k(i) = \frac{\mu_0}{\bar{\boldsymbol{x}}_k^H(i) \bar{\boldsymbol{x}}_k(i)}$.

2) Information exchange

Given the network topology structure, only the local compressed estimator $\bar{\boldsymbol{\psi}}_k(i)$ will be transmitted between node k and all its neighbor nodes. The measurement matrix $\boldsymbol{\Gamma}_k$ will be kept locally.

3) Combination

At each time instant $i = 1, 2, ..., I$, the combination step starts after the information exchange is finished. Each node will combine the local compressed estimators from its neighbor nodes and itself through

$$\bar{\boldsymbol{\omega}}_k(i+1) = \sum_{l \in \mathcal{N}_k} c_{kl} \bar{\boldsymbol{\psi}}_l(i), \tag{5.9}$$

to compute the updated compressed estimator $\bar{\boldsymbol{\omega}}_k(i+1)$.

After the final iteration I, each node will employ the OMP reconstruction strategy to generate the decompressed estimator $\boldsymbol{\omega}_k(I)$, which is illustrated in Fig. 5.1. Other

reconstruction algorithms can also be used. In summary, during the DCE procedure, only the local compressed estimator $\bar{\boldsymbol{\psi}}_k(i)$ will be transmitted over the network resulting in a reduction of the number of parameters to be transmitted from M to D. The proposed DCE scheme is given in Table 5.1.

The computational complexity of the proposed DCE scheme is $O(NDI + ND^3)$, where N is the number of nodes in the WSN and I is the number of time instants. The distributed NLMS algorithm has a complexity $O(NMI)$, while the complexity of the sparse diffusion NLMS algorithm [63] is $O(3NMI)$. For the distributed compressive sensing algorithm of [85], the computational complexity is $O(NMI + ND^3I)$. In the proposed DCE scheme, only the local compressed estimator $\bar{\boldsymbol{\psi}}_k(i)$ with D parameters will be transmitted through the network, which means the transmission requirement is greatly reduced as compared with the standard schemes that transmit $\boldsymbol{\psi}_k(i)$ with M parameters.

Table 5.1: The Proposed DCE Scheme

Initialize: $\bar{\boldsymbol{\omega}}_k(1)$=0, for $k = 1, 2, \ldots, N$

For each time instant $i = 1, 2, ..., I-1$

  For each node $k = 1, 2, \ldots, N$

    $\bar{\boldsymbol{\psi}}_k(i) = \bar{\boldsymbol{\omega}}_k(i) + \mu(i) e_k^*(i) \bar{\boldsymbol{x}}_k(i)$

    where $e_k(i) = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i) \bar{\boldsymbol{x}}_k(i)$,

  For each node $k = 1, 2, \ldots, N$

    $\bar{\boldsymbol{\omega}}_k(i+1) = \sum_{l \in \mathcal{N}_k} c_{kl} \bar{\boldsymbol{\psi}}_l(i)$

  end

end

After the final iteration I

For each node $k = 1, 2, \ldots, N$

  $\boldsymbol{\omega}_k(I) = f_{\text{OMP}}\{\bar{\boldsymbol{\omega}}_k(I)\}$ % using OMP to decompress the estimator

  where $\boldsymbol{\omega}_k(I)$ is the final decompressed estimator.

end

5.4 Measurement Matrix Optimization

To further improve the performance of the proposed DCE scheme, an optimization algorithm for the design of the measurement matrix $\mathbf{\Gamma}_k(i)$, which is now time–variant, is developed here. Unlike prior work [84, 89], this optimization is distributed and adaptive. Let us consider the cost function

$$\begin{aligned}\mathcal{J} &= \mathbb{E}\{|e_k(i)|^2\} \\ &= \mathbb{E}\{|d_k(i) - y_k(i)|^2\} \\ &= \mathbb{E}\{|d_k(i)|^2\} - \mathbb{E}\{d_k^*(i)y_k(i)\} - \mathbb{E}\{d_k(i)y_k^*(i)\} + \mathbb{E}\{|y_k(i)|^2\},\end{aligned} \tag{5.10}$$

where $y_k(i) = \bar{\boldsymbol{\omega}}_k^H(i)\bar{\boldsymbol{x}}_k(i)$. To minimize the cost function, we need to compute the gradient of $\mathcal{J}$ with respect to $\mathbf{\Gamma}_k^*(i)$ and equate it to a null vector, i.e., $\nabla\mathcal{J}_{\mathbf{\Gamma}_k^*(i)} = \mathbf{0}$. As a result, only the first three terms in (5.10) need to be considered. Taking the first three terms of (5.10) we arrive at

$$\begin{aligned}&\mathbb{E}\{|d_k(i)|^2\} - \mathbb{E}\{d_k^*(i)y_k(i)\} - \mathbb{E}\{d_k(i)y_k^*(i)\} \\ &= \mathbb{E}\{|\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i) + n_k(i)|^2\} - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i) + n_k(i))^* y_k(i)\} \\ &\quad - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i) + n_k(i))y_k^*(i)\} \\ &= \mathbb{E}\{|\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i)|^2\} + \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i))^* n_k(i)\} + \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i))n_k^*(i)\} \\ &\quad + \mathbb{E}\{|n_k(i)|^2\} - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i))^* y_k(i)\} - \mathbb{E}\{n_k^*(i)y_k(i)\} \\ &\quad - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i))y_k^*(i)\} - \mathbb{E}\{n_k(i)y_k^*(i)\}.\end{aligned} \tag{5.11}$$

Because the random variable $n_k(i)$ is statistically independent from the other parameters and has zero mean, (5.11) can be further simplified as

$$\begin{aligned}&\mathbb{E}\{|d_k(i)|^2\} - \mathbb{E}\{d_k^*(i)y_k(i)\} - \mathbb{E}\{d_k(i)y_k^*(i)\} \\ &= \mathbb{E}\{|\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i)|^2\} + \sigma_{n,k}^2 - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\bar{\boldsymbol{x}}_k(i))^* y_k(i)\} - \mathbb{E}\{(\boldsymbol{\omega}_0^H\mathbf{\Gamma}_k^H(i)\boldsymbol{x}_k(i))y_k^*(i)\}.\end{aligned} \tag{5.12}$$

Then, we have

$$\nabla\mathcal{J}_{\mathbf{\Gamma}_k^*(i)} = \boldsymbol{R}_k(i)\mathbf{\Gamma}_k(i)\boldsymbol{R}_{\boldsymbol{\omega}_0} - \boldsymbol{P}_k(i), \tag{5.13}$$

where $\boldsymbol{R}_k(i) = \mathbb{E}\{\bar{\boldsymbol{x}}_k(i)\bar{\boldsymbol{x}}_k^H(i)\}$, $\boldsymbol{R}_{\boldsymbol{\omega}_0} = \mathbb{E}\{\boldsymbol{\omega}_0\boldsymbol{\omega}_0^H\}$ and $\boldsymbol{P}_k(i) = \mathbb{E}\{y_k^*(i)\bar{\boldsymbol{x}}_k(i)\boldsymbol{\omega}_0^H\}$. Equating (5.13) to a null vector, we obtain

$$\boldsymbol{R}_k(i)\mathbf{\Gamma}_k(i)\boldsymbol{R}_{\boldsymbol{\omega}_0} - \boldsymbol{P}_k(i) = \mathbf{0}, \tag{5.14}$$

$$\boldsymbol{\Gamma}_k(i) = \boldsymbol{R}_k^{-1}(i)\boldsymbol{P}_k(i)\boldsymbol{R}_{\boldsymbol{\omega}_0}^{-1}. \tag{5.15}$$

The expression in (5.15) cannot be solved in closed–form because $\boldsymbol{\omega}_0$ is an unknown parameter. As a result, we employ the previous estimate $\bar{\boldsymbol{\omega}}_k(i)$ to replace $\boldsymbol{\omega}_0$. However, $\bar{\boldsymbol{\omega}}_k(i)$ and $\boldsymbol{\Gamma}_k(i)$ depend on each other, thus, it is necessary to iterate (5.15) with an initial guess to obtain a solution. In particular, we replace the expected values with instantaneous values. Starting from (5.13), we use instantaneous estimates to compute

$$\hat{\boldsymbol{R}}_k(i) = \bar{\boldsymbol{x}}_k(i)\bar{\boldsymbol{x}}_k^H(i), \tag{5.16}$$

$$\hat{\boldsymbol{R}}_{\boldsymbol{\omega}_0} = \boldsymbol{\omega}_0\boldsymbol{\omega}_0^H \tag{5.17}$$

and

$$\hat{\boldsymbol{P}}_k(i) = y_k^*(i)\bar{\boldsymbol{x}}_k(i)\boldsymbol{\omega}_0^H. \tag{5.18}$$

According to the method of steepest descent [33], the updated parameters of the measurement matrix $\boldsymbol{\Gamma}_k(i)$ at time $i+1$ are computed by using the simple recursive relation

$$\begin{aligned}\boldsymbol{\Gamma}_k(i+1) &= \boldsymbol{\Gamma}_k(i) + \eta[-\nabla\mathcal{J}_{\boldsymbol{\Gamma}_k^*(i)}] \\ &= \boldsymbol{\Gamma}_k(i) + \eta[\hat{\boldsymbol{P}}_k(i) - \hat{\boldsymbol{R}}_k(i)\boldsymbol{\Gamma}_k(i)\hat{\boldsymbol{R}}_{\boldsymbol{\omega}_0}] \\ &= \boldsymbol{\Gamma}_k(i) + \eta[y_k^*(i)\bar{\boldsymbol{x}}_k(i)\boldsymbol{\omega}_0^H - \bar{\boldsymbol{x}}_k(i)\bar{\boldsymbol{x}}_k^H(i)\boldsymbol{\Gamma}_k(i)\boldsymbol{\omega}_0\boldsymbol{\omega}_0^H].\end{aligned} \tag{5.19}$$

where η is the step size and $\boldsymbol{\omega}_0$ is the $M \times 1$ unknown parameter vector that must be estimated by the network. Then, the parameter vector $\bar{\boldsymbol{\omega}}_k(i)$ is used to reconstruct the estimate of $\boldsymbol{\omega}_0$ as follows

$$\boldsymbol{\omega}_{re_k}(i) = f_{\text{OMP}}\{\bar{\boldsymbol{\omega}}_k(i)\}, \tag{5.20}$$

where the operator $f_{\text{OMP}}\{\cdot\}$ denotes the OMP reconstruction algorithm described in Chapter 2 and $\boldsymbol{\omega}_{re_k}(i)$ is the decompressed estimator from $\bar{\boldsymbol{\omega}}_k(i)$. Note that other reconstruction algorithms could also be employed. Replacing $\boldsymbol{\omega}_0$ by $\boldsymbol{\omega}_{re_k}(i)$, we arrive at the expression for updating the measurement matrix described by

$$\boldsymbol{\Gamma}_k(i+1) = \boldsymbol{\Gamma}_k(i) + \eta\big[y_k^*(i)\bar{\boldsymbol{x}}_k(i)\boldsymbol{\omega}_{re_k}^H(i) - \bar{\boldsymbol{x}}_k(i)\bar{\boldsymbol{x}}_k^H(i)\boldsymbol{\Gamma}_k(i)\boldsymbol{\omega}_{re_k}(i)\boldsymbol{\omega}_{re_k}^H(i)\big]. \tag{5.21}$$

The computational complexity of the proposed scheme with measurement matrix optimization is $O(NDI + ND^3I)$.

5.5 Simulations

We assess the proposed DCE scheme and the measurement matrix optimization algorithm in a WSN application, where a partially connected network with N = 20 nodes is considered and illustrated in Fig. 5.3. We compare the proposed DCE scheme with uncompressed schemes, including the distributed NLMS (dNLMS) algorithm (normalized version of [2]), sparse diffusion NLMS algorithm [63], sparsity-promoting adaptive algorithm [78], and the distributed compressive sensing algorithm [85], in terms of MSE performance. Note that other metrics such as mean-square deviation (MSD) could be used but result in the same performance hierarchy between the analyzed algorithms.

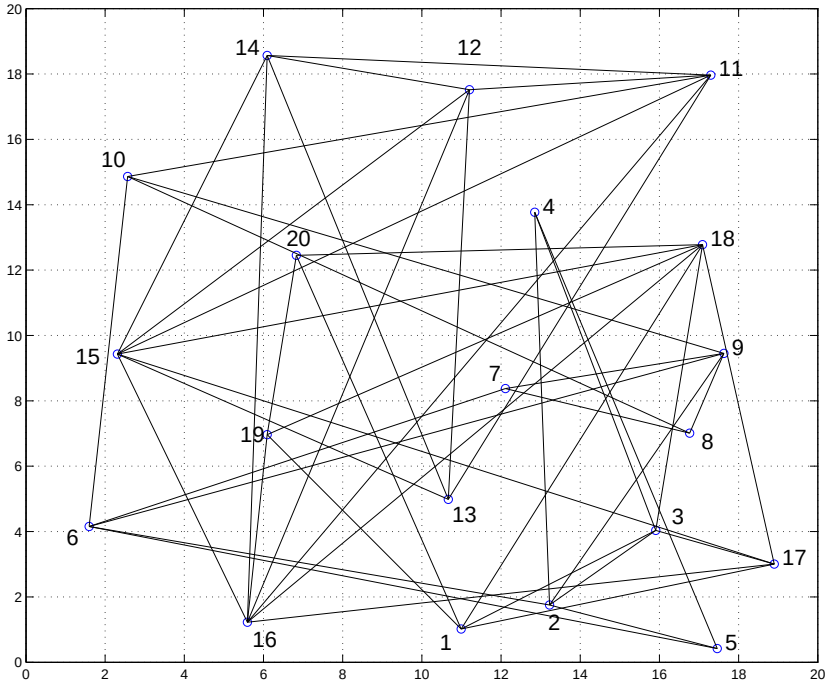

Figure 5.3: Diffusion wireless sensor network with 20 Nodes

The input signal is generated as $\boldsymbol{x}_k(i) = [x_k(i) \quad x_k(i-1) \quad ... \quad x_k(i-M+1)]^T$ and $x_k(i) = u_k(i)+\alpha_k x_k(i-1)$, where α_k is a correlation coefficient and $u_k(i)$ is a white noise process with variance $\sigma_{u,k}^2 = 1 - |\alpha_k|^2$, to ensure the variance of $\boldsymbol{x}_k(i)$ is $\sigma_{x,k}^2 = 1$ [1]. The compressed input signal is obtained by $\bar{\boldsymbol{x}}_k(i) = \boldsymbol{\Gamma}_k \boldsymbol{x}_k(i)$. The measurement matrix $\boldsymbol{\Gamma}_k$ is an i.i.d. Gaussian random matrix that is kept constant. The noise samples are

modeled as complex Gaussian noise with variance $\sigma_{n,k}^2 = 0.001$. The unknown $M \times 1$ parameter vector $\boldsymbol{\omega}_0$, which is generated randomly, has sparsity S, where M=50, D=10 and S=3. The step size μ_0 for the distributed NLMS, distributed compressive sensing, sparse diffusion LMS and the proposed DCE algorithms is 0.15. The parameter that controls the shrinkage in [63] is set to 0.001. For [78], the number of hyperslabs equals 55 and the width of the hyperslabs is 0.01. The parameter η for the measurement matrix optimization algorithm is set to 0.08.

Fig. 5.4 illustrates the comparison between the DCE scheme with other existing algorithms, together with the measurement matrix optimization. It is clear that, when compared with the existing algorithms, the DCE scheme has a significantly faster convergence rate and a better MSE performance, and with the help of the measurement matrix optimization algorithm, DCE can achieve a faster convergence when compared with DCE without the measurement matrix optimization.

These advantages consist in two features: the compressed dimension brought by the proposed scheme and CS being implemented in the estimation layer. As a result, the number of parameters for transmission in the network is significantly reduced.

In the second scenario, we illustrate the DCE scheme with different levels of resolution in bits per coefficient, reduced dimension D and sparsity level S. The x-axis stands for the reduced dimension D and their corresponding sparsity level S can be found in Fig. 5.5. In Fig. 5.5, it is clear that with the increase of the sparsity level S the MSE performance degrades. In addition, the MSE performance will increase when the transmission has more bits per coefficient. For the DCE scheme, the total number of bits required for transmission is D times the number of bits per coefficient. A certain level of redundancy is required between the sparsity level and the reduced dimension due to the error introduced by the estimation procedure.

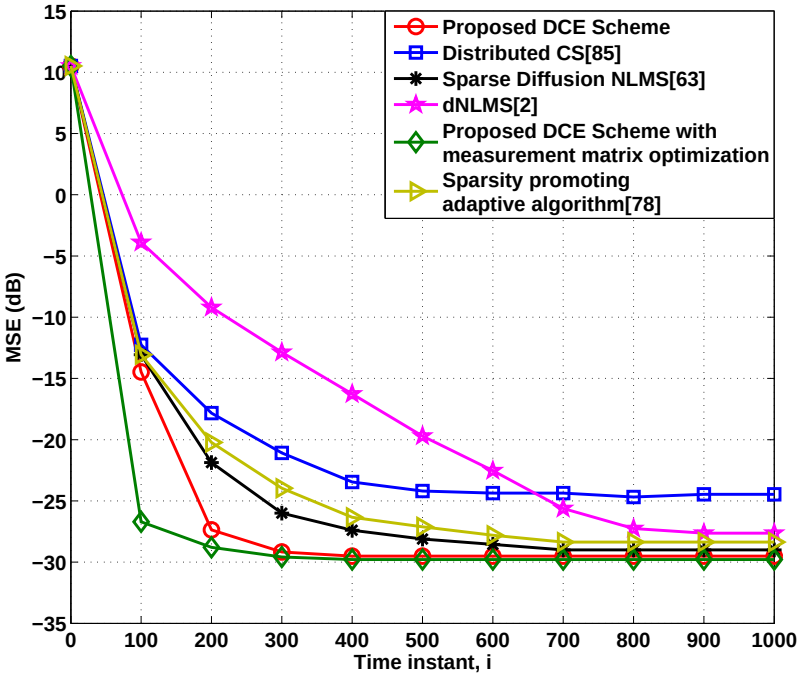

Figure 5.4: MSE performance against time

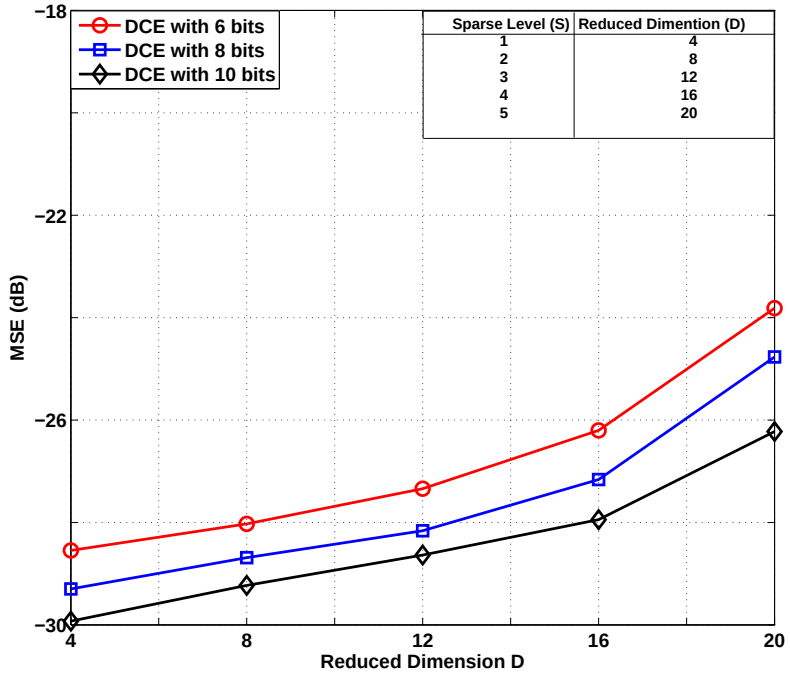

Figure 5.5: MSE performance against reduced dimension D for different levels of resolution in bits per coefficient

5.6 Summary

In this chapter, we have proposed a novel DCE scheme and algorithms for sparse signals and systems based on CS techniques and a measurement matrix optimization strategy. In the DCE scheme, the estimation procedure is performed in a compressed dimension. Meanwhile, the proposed measurement matrix optimization strategy can further improve the performance of the proposed DCE scheme. The simulation results for a WSN application show that the DCE scheme outperforms existing strategies in terms of convergence rate, reduced bandwidth and MSE performance.

Chapter 6

Distributed Reduced–Rank Estimation Based on Joint Iterative Optimization in Sensor Networks

Contents

6.1 Introduction 114

6.2 System Model 116

6.3 Distributed Dimensionality Reduction and Adaptive Processing . . 118

6.4 Proposed Distributed Reduced-Rank Algorithms 119

6.5 Simulation results 126

6.6 Summary 134

6.1 Introduction

Distributed strategies have become fundamental for parameter estimation in wireless networks and applications such as sensor networks [1, 2] and smart grids [4, 29]. Distributed processing techniques deal with the extraction of information from data collected at nodes that are distributed over a geographic area [1]. In this context, a specific node or agent in the network collects data from its neighbors and combines them with its local information

to generate an improved estimate. However, when the unknown parameter vector to be estimated has a large dimension, the network requires a large communication bandwidth between neighbor nodes to transmit their local estimate. This problem limits the application of existing algorithms in applications with large data sets as the convergence speed is dependent on the length of the parameter vector [2, 9, 10]. Hence, distributed dimensionality reduction has become an important tool for distributed inference problems with large data sets.

In order to perform dimensionality reduction, many algorithms have been proposed in the literature, in the context of distributed quantized Kalman Filtering [90, 91], quantized consensus algorithms [92], distributed principal subspace estimation [93], single bit strategy [94] and Krylov subspace optimization techniques [95]. However, existing algorithms are either too costly or have unsatisfactory performance when processing a large number of parameters. As a result, trade-offs between the amount of cooperation, communication and system performance naturally exist. In this context, reduced–rank techniques are powerful tools to perform dimensionality reduction, which have been applied to DS–CDMA system [96–98], multi–input–multi–output (MIMO) equalization application [99], spread–spectrum systems [100], space–time interference suppression [13] and beamforming [101, 102]. However, limited research has been carried out on distributed reduced-rank estimation. Related approaches to reduced–rank techniques include compressive sensing-based strategies [84], which exploit sparsity to reduce the number of parameters for estimation, and attribute-distributed learning [103], which employs agents and a fusion center to meet the communication constraints.

In this chapter, we propose a scheme for distributed signal processing along with distributed reduced–rank algorithms for parameter estimation. In particular, the proposed algorithms are based on an alternating optimization strategy [96, 99] and are called distributed reduced–rank joint iterative optimization normalized least mean squares (DRJIO–NLMS) algorithm and distributed reduced–rank joint iterative optimization recursive least square (DRJIO–RLS) algorithm. In contrast to prior work on reduced–rank techniques and distributed methods, this is to the best of our knowledge the first time this alternating optimization strategy is done in a distributed way. The proposed reduced–rank strategies are distributed and perform dimensionality reduction without costly decompositions at each agent. The proposed DRJIO–NLMS and DRJIO–RLS algorithms are flexible with

regards to the amount of information that is exchanged, have low cost and high performance. The DRJIO–NLMS and DRJIO–RLS algorithms can also outperform competing techniques. We also present a computational complexity analysis of the proposed and existing reduced–rank algorithms. Applications to parameter estimation in wireless sensor networks and smart grids are then studied.

6.2 System Model

A distributed network with N nodes, which have limited processing capabilities, is considered with a partially connected topology. A diffusion protocol is employed although other strategies, such as incremental [1] and consensus-based [4] could also be used. A partially connected network means that nodes can exchange information only with their neighbors determined by the connectivity topology. In contrast, a fully connected network means that, data broadcast by a node can be captured by all other nodes in the network [19]. At every time instant i, each node k takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = \boldsymbol{\omega}_0^H \boldsymbol{x}_k(i) + n_k(i), \quad i = 1, 2, \ldots, \mathrm{I}, \tag{6.1}$$

where $\boldsymbol{x}_k(i)$ is the $M \times 1$ input signal vector with zero mean and variance $\sigma_{x,k}^2$, $n_k(i)$ is the noise sample at each node with zero mean and variance $\sigma_{n,k}^2$. Through (6.1), we can see that the measurements for all nodes are related to an unknown parameter vector $\boldsymbol{\omega}_0$ with size $M \times 1$, that would be estimated by the network. Fig. 6.1 shows an example for a diffusion–type network with 20 nodes. The aim of such a network is to compute an estimate of $\boldsymbol{\omega}_0$ in a distributed fashion, which can minimize the global cost function

$$J_\omega(\boldsymbol{\omega}) = \sum_{k=1}^{N} \mathbb{E}\big|d_k(i) - \boldsymbol{\omega}^H \boldsymbol{x}_k(i)\big|^2, \tag{6.2}$$

where $\mathbb{E}$ denotes the expectation operator. To solve this problem, one possible technique is the adapt–then–combine (ATC) diffusion strategy [2]

$$\begin{cases} \boldsymbol{\psi}_k(i) = \boldsymbol{\omega}_k(i-1) + \mu_k \boldsymbol{x}_k(i)\big[d_k(i) - \boldsymbol{\omega}_k^H(i-1)\boldsymbol{x}_k(i)\big]^*, \\ \\ \boldsymbol{\omega}_k(i) = \sum\limits_{l \in \mathcal{N}_k} c_{kl} \boldsymbol{\psi}_l(i), \end{cases} \tag{6.3}$$

where $\mathcal{N}_k$ indicates the set of neighbors for node k, $|\mathcal{N}_k|$ denotes the cardinality of $\mathcal{N}_k$ and c_{kl} is the combination coefficient, which is calculated under the Metropolis rule

$$\begin{cases} c_{kl} = \frac{1}{max(|\mathcal{N}_k|,|\mathcal{N}_l|)}, & \text{if } k \neq l \text{ are linked} \\ c_{kl} = 0, & \text{for } k \text{ and } l \text{ not linked} \\ c_{kk} = 1 - \sum\limits_{l \in \mathcal{N}_k/k} c_{kl}, & \text{for } k = l \end{cases} \tag{6.4}$$

and should satisfy

$$\sum_{l} c_{kl} = 1, l \in \mathcal{N}_k \forall k. \tag{6.5}$$

With this strategy, when the dimension of the unknown parameter vector $\boldsymbol{\omega}_0$ is large, it could lead to a high communication overhead between each neighbor node and the convergence speed is reduced. In order to solve this problem and optimize the distributed processing, we incorporate at the kth node of the network distributed reduced–rank strategies based on alternating optimization techniques.

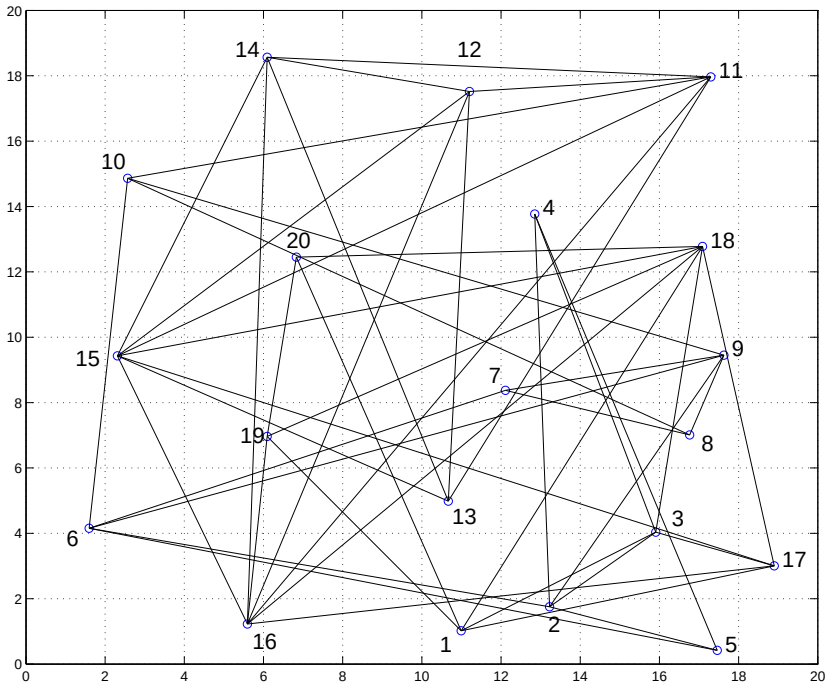

Figure 6.1: Network topology with 20 nodes

6.3 Distributed Dimensionality Reduction and Adaptive Processing

The proposed distributed dimensionality reduction scheme, depicted in Fig.6.2, employs a transformation matrix $\boldsymbol{S}_{D_k}(i)$ to process the input signal $\boldsymbol{x}_k(i)$ with dimensions $M \times 1$ and projects it onto a lower $D \times 1$ dimensional subspace $\bar{\boldsymbol{x}}_k(i)$, where $D \ll M$. Following this procedure, a reduced–rank estimator $\bar{\boldsymbol{\omega}}_k(i)$ is computed, and the $\bar{\boldsymbol{\omega}}_k(i)$ is transmitted by each node. In particular, the transformation matrix $\boldsymbol{S}_{D_k}(i)$ and reduced–rank estimator $\bar{\boldsymbol{\omega}}_k(i)$ will be jointly optimized in the proposed scheme according to the minimum mean–squared error (MMSE) criterion.

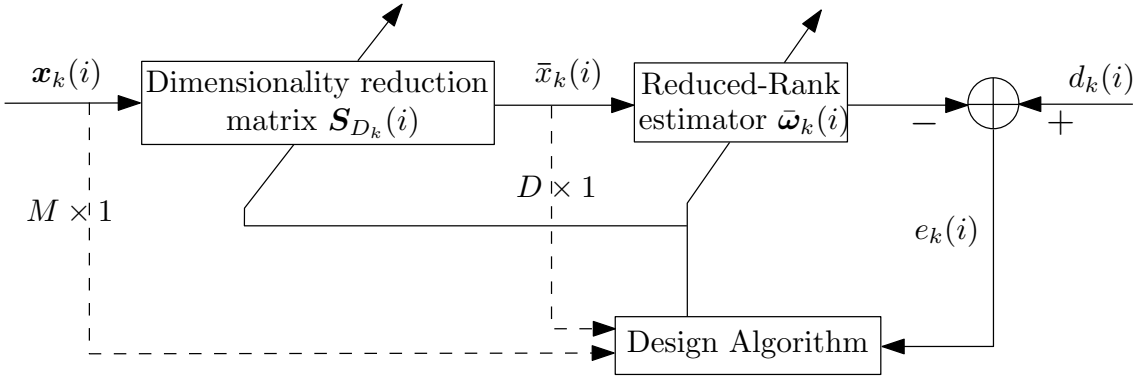

Figure 6.2: Proposed dimensionality reduction scheme at each node or agent

Specifically, we start the description of the method with an $M \times D$ matrix $\boldsymbol{S}_{D_k}(i)$, which carries out a dimensionality reduction on the input signal as given by

$$\bar{\boldsymbol{x}}_k(i) = \boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i), \tag{6.6}$$

where, in what follows, all D–dimensional quantities are denoted with a 'bar'. The design of $\boldsymbol{S}_{D_k}(i)$ and $\bar{\boldsymbol{\omega}}_k(i)$ corresponds to the optimization problem given by

$$\left\{\boldsymbol{S}_{D_k}^{\text{opt}}, \bar{\boldsymbol{\omega}}_k^{\text{opt}}\right\} = \min_{\boldsymbol{S}_{D_k}(i), \bar{\boldsymbol{\omega}}_k(i)} \mathbb{E}[|d_k(i) - \bar{\boldsymbol{\omega}}_k{}^H(i)\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i)|^2] \tag{6.7}$$

where $\bar{\boldsymbol{\omega}}_k(i)$ is the reduced–rank estimator. By fixing $\boldsymbol{S}_{D_k}(i)$ and minimizing (6.7) with respect to $\bar{\boldsymbol{\omega}}_k(i)$, we have

$$\bar{\boldsymbol{\omega}}_k(i) = \bar{\boldsymbol{R}}_k^{-1}(i)\bar{\boldsymbol{p}}_k(i), \tag{6.8}$$

where $\bar{\boldsymbol{R}}_k(i) = \mathbb{E}[\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{S}_{D_k}(i)] = \mathbb{E}[\bar{\boldsymbol{x}}_k(i)\bar{\boldsymbol{x}}_k^H(i)]$ and $\bar{\boldsymbol{p}}_k(i) = \mathbb{E}[d_k^*(i)\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i)] = \mathbb{E}[d_k^*(i)\bar{\boldsymbol{x}}_k(i)]$. We then fix $\bar{\boldsymbol{\omega}}_k(i)$ and minimize (6.7) with respect to $\boldsymbol{S}_{D_k}(i)$, and arrive at the following expression:

$$\boldsymbol{S}_{D_k}(i) = \boldsymbol{R}_k^{-1}(i)\boldsymbol{P}_{D_k}(i)\bar{\boldsymbol{R}}_{\bar{\boldsymbol{\omega}}_k}^{-1}(i), \tag{6.9}$$

where $\boldsymbol{R}_k(i) = \mathbb{E}[\boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)]$, $\boldsymbol{P}_{D_k}(i) = \mathbb{E}[d_k^*(i)\boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k^H(i)]$ and $\bar{\boldsymbol{R}}_{\bar{\boldsymbol{\omega}}_k}(i) = \mathbb{E}[\bar{\boldsymbol{\omega}}_k(i)\bar{\boldsymbol{\omega}}_k^H(i)]$. The associated reduced–rank MSE is obtained by substituting the expressions obtained in (6.8) and (6.9) into the cost function and is described as [96]

$$\mathbf{MSE} = \sigma_{d_k}^2 - \bar{\boldsymbol{p}}_k^H(i)\bar{\boldsymbol{R}}_k^{-1}(i)\bar{\boldsymbol{p}}_k(i) \tag{6.10}$$

where $\sigma_{d_k}^2 = \mathbb{E}[|d_k(i)|^2]$. Because there is no closed-form expression for $\boldsymbol{S}_{D_k}(i)$ and $\bar{\boldsymbol{\omega}}_k(i)$ as they depend on each others, we need a strategy to compute the parameters. The proposed strategy is based on an alternating optimization of $\boldsymbol{S}_{D_k}(i)$ and $\bar{\boldsymbol{\omega}}_k(i)$. In the next section, we develop a distributed reduced-rank algorithm to compute the parameters of interest.

6.4 Proposed Distributed Reduced-Rank Algorithms

In this section, we present the two proposed distributed reduced–rank algorithms for distributed estimation, namely DRJIO–NLMS and DRJIO–RLS. Unlike prior work [93–95], the proposed algorithms do **NOT** require

- an $M \times M$ auto–correlation matrix of the input signal and an $M \times 1$ cross–correlation vector between the input signal and the desired signal used to build the Krylov subspace [95]
- Additional cost to perform eigen–decompositions [93]
- Extra adaptive processing at the local node [94]
- Costly convex optimization at the local node, which introduces extra complexity [95].

The DRJIO–NLMS and DRJIO–RLS algorithms are flexible, have low cost and very fast convergence speed.

6.4.1 Proposed DRJIO–NLMS algorithm

For the DRJIO–NLMS algorithm, the parameters in (6.7) are optimized by an alternating procedure that adjusts one of the parameters while keeping the other parameter fixed. Therefore, it solves the following optimization problem in an alternating fashion:

$$\begin{aligned}\left[\boldsymbol{S}_{D_k}^{\text{opt}}, \bar{\boldsymbol{\omega}}_k^{\text{opt}}\right] &= \min_{\boldsymbol{S}_{D_k}(i), \bar{\boldsymbol{\omega}}_k^H(i)} ||\bar{\boldsymbol{\omega}}_k(i) - \bar{\boldsymbol{\omega}}_k(i-1)||^2 + ||\boldsymbol{S}_{D_k}(i) - \boldsymbol{S}_{D_k}(i-1)||^2 \\ &\text{subject to } \bar{\boldsymbol{\omega}}_k^H(i)\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i) = d_k(i).\end{aligned} \tag{6.11}$$

Using the method of Lagrange multipliers and considering $\{\boldsymbol{S}_{D_k}, \bar{\boldsymbol{\omega}}_k\}$ jointly, we arrive at the following Lagrangian:

$$\begin{aligned}\mathcal{L}_k = &||\bar{\boldsymbol{\omega}}_k(i) - \bar{\boldsymbol{\omega}}_k(i-1)||^2 + ||\boldsymbol{S}_{D_k}(i) - \boldsymbol{S}_{D_k}(i-1)||^2 \\ &+ \Re[\lambda_1^*(d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i)\boldsymbol{S}_{D_k}^H(i-1)\boldsymbol{x}_k(i))] \\ &+ \Re[\lambda_2^*(d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i))],\end{aligned} \tag{6.12}$$

where , λ_1, λ_2 are scalar Lagrange multipliers, $||\cdot||$ denotes the Frobenius norm, and the operator $\Re[\cdot]$ retains the real part of the argument. By computing the gradient terms of (6.12) with respect to $\bar{\boldsymbol{\omega}}_k(i)$, $\boldsymbol{S}_{D_k}(i)$, λ_1 and λ_2, respectively, we obtain

$$\nabla_{\bar{\boldsymbol{\omega}}_k(i)}\mathcal{L} = 2\big(\bar{\boldsymbol{\omega}}_k(i) - \bar{\boldsymbol{\omega}}_k(i-1)\big) + \boldsymbol{S}_{D_k}^H(i-1)\boldsymbol{x}_k(i)\lambda_1 \tag{6.13}$$

$$\nabla_{\boldsymbol{S}_{D_k}(i)}\mathcal{L} = 2\big(\boldsymbol{S}_{D_k}(i) - \boldsymbol{S}_{D_k}(i-1)\big) + \boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k(i-1)\lambda_2 \tag{6.14}$$

$$\nabla_{\lambda_1}\mathcal{L} = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i)\boldsymbol{S}_{D_k}^H(i-1)\boldsymbol{x}_k(i) \tag{6.15}$$

$$\nabla_{\lambda_2}\mathcal{L} = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{S}_{D_k}^H(i)\boldsymbol{x}_k(i). \tag{6.16}$$

By setting (6.13)–(6.16) to zero and solving the remaining equations, we obtain the recursions of the proposed DRJIO–NLMS algorithm described by

$$\bar{\boldsymbol{\omega}}_k(i) = \bar{\boldsymbol{\omega}}_k(i-1) + \mu(i)e_k^*(i)\bar{\boldsymbol{x}}_k(i) \tag{6.17}$$

$$\boldsymbol{S}_{D_k}(i) = \boldsymbol{S}_{D_k}(i-1) + \eta(i)e_k^*(i)\boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k^H(i-1) \tag{6.18}$$

where $e_k(i) = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{S}_{D_k}^H(i-1)\boldsymbol{x}_k(i)$, $\mu(i) = \frac{\mu_0}{\boldsymbol{x}_k^H(i)\boldsymbol{x}_k(i)}$ and $\eta(i) = \frac{\eta_0}{\bar{\boldsymbol{\omega}}_k^H(i-1)\bar{\boldsymbol{\omega}}_k(i-1)\boldsymbol{x}_k^H(i)\boldsymbol{x}_k(i)}$ are the time–varying step sizes, while μ_0 and η_0 are the convergence factors. The normalization makes it easier the setting of the convergence factors, improves the convergence speed and facilitates the comparison with other distributed

LMS–type algorithms. The recursions are computed in an alternating way with one iteration per time instant at each node.

The proposed DRJIO–NLMS algorithm includes two steps, namely, adaptation step and combination step, which are performed using an alternating procedure which is detailed next.

- Adaptation step

For the adaptation step, at each time instant i=1,2, . . . , I, each node k=1,2, . . . , N, starts from generating a local reduced–rank estimator through

$$\bar{\boldsymbol{\psi}}_k(i) = \bar{\boldsymbol{\omega}}_k(i-1) + \mu(i)e_k^*(i)\bar{\boldsymbol{x}}_k(i), \tag{6.19}$$

where $e_k(i) = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{S}_{D_k}^H(i-1)\boldsymbol{x}_k(i)$. This local reduced–rank estimator $\bar{\boldsymbol{\psi}}_k(i)$ will be transmitted to all its neighbor nodes under the network topology structure.

Then, each node k=1,2, . . . , N, will update its dimensionality reduction matrix according to

$$\boldsymbol{S}_{D_k}(i) = \boldsymbol{S}_{D_k}(i-1) + \eta(i)e_k^*(i)\boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k(i-1), \tag{6.20}$$

and keep it locally.

- Combination step

At each time instant i=1,2, . . . , I, the combination step starts after the adaptation step finishes. Each node will combine the local reduced–rank estimators from its neighbor nodes and itself through

$$\bar{\boldsymbol{\omega}}_k(i) = \sum_{l\in\mathcal{N}_k} c_{kl}\bar{\boldsymbol{\psi}}_l(i), \tag{6.21}$$

to compute the reduced–rank estimator $\bar{\boldsymbol{\omega}}_k(i)$.

After the final iteration I, each node will generate the full–rank estimator $\boldsymbol{\omega}_k(I)$ from

$$\boldsymbol{\omega}_k(I) = \boldsymbol{S}_{D_k}(I)\bar{\boldsymbol{\omega}}_k(I). \tag{6.22}$$

In conclusion, during the distributed processing steps, only the local reduced–rank estimator $\bar{\boldsymbol{\psi}}_k(i)$ will be transmitted through the network. The proposed DRJIO–NLMS algorithm is detailed in Table 6.1.

Table 6.1: The DRJIO–NLMS Algorithm

Initialize: $\bar{\boldsymbol{\omega}}_k(0)$=0

$\boldsymbol{S}_{D_k}^H(0) = [\boldsymbol{I}_D \;\; \boldsymbol{0}_{D,M-D}]^T, k = 1, 2, ..., N$

For each time instant $i = 1, 2, ..., I$

  For each node $k = 1, 2, \ldots, N$

    $\bar{\boldsymbol{\psi}}_k(i) = \bar{\boldsymbol{\omega}}_k(i-1) + \mu(i) e_k^*(i) \bar{\boldsymbol{x}}_k(i)$

    where $e_k(i) = d_k(i) - \bar{\boldsymbol{\omega}}_k^H(i-1) \boldsymbol{S}_{D_k}^H(i-1) \boldsymbol{x}_k(i)$

    % $\bar{\boldsymbol{\psi}}_k(i)$ is the local reduced–rank estimator and will be

    % sent to all neighbor nodes of node k under the network

    % topology structure.

    $\boldsymbol{S}_{D_k}(i) = \boldsymbol{S}_{D_k}(i-1) + \eta(i) e_k^*(i) \boldsymbol{x}_k(i) \bar{\boldsymbol{\omega}}_k(i-1)$

    % The dimensionality reduction matrix $\boldsymbol{S}_{D_k}(i)$

    % will be updated and kept locally.

  end

  For each node $k = 1, 2, \ldots, N$

    $\bar{\boldsymbol{\omega}}_k(i) = \sum_{l \in \mathcal{N}_k} c_{kl} \bar{\boldsymbol{\psi}}_l(i)$

    % The reduced–rank estimator $\bar{\boldsymbol{\omega}}_k(i)$

    % will be updated and kept locally.

  end

end

After the final iteration I

For each node $k = 1, 2, \ldots, N$

  $\boldsymbol{\omega}_k(I) = \boldsymbol{S}_{D_k}(I) \bar{\boldsymbol{\omega}}_k(I)$

  where $\boldsymbol{\omega}_k(I)$ is the final full–rank estimator.

end

6.4.2 Proposed DRJIO–RLS algorithm

In this subsection, we present a recursive approach for the dimensionality reduction distributed estimation. Specifically, we develop the DRJIO–RLS algorithm for computing $\boldsymbol{S}_{D_k}(i)$ and $\bar{\boldsymbol{\omega}}_k(i)$. In order to derive the proposed algorithm, we first define

$$\boldsymbol{P}_k(i) \triangleq \tilde{\boldsymbol{R}}_k^{-1}(i) \tag{6.23}$$

where $\tilde{\boldsymbol{R}}_k(i) = \sum_{l=1}^{i} \lambda^{i-l} \boldsymbol{x}_k(i)\boldsymbol{x}_k^H(i)$,

$$\boldsymbol{P}_{D_k}(i) \triangleq \lambda \boldsymbol{P}_{D_k}(i-1) + d_k^*(i)\boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k^H(i) \tag{6.24}$$

$$\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) \triangleq \bar{\boldsymbol{R}}_{\bar{\boldsymbol{\omega}}_k}^{-1}(i-1) \tag{6.25}$$

and rewrite the expression in (6.9) as follows

$$\begin{aligned}
\boldsymbol{S}_{D_k}(i) &= \tilde{\boldsymbol{R}}_k^{-1}(i)\boldsymbol{P}_{D_k}(i)\bar{\boldsymbol{R}}_{\bar{\boldsymbol{\omega}}_k}^{-1}(i-1) \\
&= \boldsymbol{P}_k(i)\boldsymbol{P}_{D_k}(i)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) \\
&= \lambda \boldsymbol{P}_k(i)\boldsymbol{P}_{D_k}(i-1)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) + d_k^*(i)\boldsymbol{P}_k(i)\boldsymbol{x}_k(i)\bar{\boldsymbol{\omega}}_k^H(i)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) \\
&= \boldsymbol{S}_{D_k}(i-1) + \boldsymbol{k}_k(i)\Big[d_k^*(i)\boldsymbol{t}_k^H(i) - \boldsymbol{x}_k^H(i)\boldsymbol{S}_{D_k}(i-1)\Big],
\end{aligned} \tag{6.26}$$

where the $D \times 1$ vector $\boldsymbol{t}_k(i) = \boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i)\bar{\boldsymbol{\omega}}_k(i)$ and the $M \times 1$ Kalman gain vector is

$$\boldsymbol{k}_k(i) = \frac{\lambda^{-1}\boldsymbol{P}_k(i-1)\boldsymbol{x}_k(i)}{1 + \lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i-1)\boldsymbol{x}_k(i)}. \tag{6.27}$$

In addition, the update for the $M \times M$ matrix $\boldsymbol{P}_k(i)$ employs the matrix inversion lemma [9] as follows:

$$\boldsymbol{P}_k(i) = \lambda^{-1}\boldsymbol{P}_k(i-1) - \lambda^{-1}\boldsymbol{k}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i-1) \tag{6.28}$$

and the $D \times 1$ vector $\boldsymbol{t}_k(i)$ is updated as

$$\boldsymbol{t}_k(i) = \frac{\lambda^{-1}\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1)\bar{\boldsymbol{\omega}}_k(i-1)}{1 + \lambda^{-1}\bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1)\bar{\boldsymbol{\omega}}_k(i-1)}. \tag{6.29}$$

The matrix inversion lemma [9] is then used to update the $D \times D$ matrix $\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i)$ as described by

$$\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) = \lambda^{-1}\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1) - \lambda^{-1}\boldsymbol{t}_k(i)\bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1). \tag{6.30}$$

Equations (6.23)–(6.30) constitute the key steps of the proposed DRJIO-RLS algorithm for computing $\boldsymbol{S}_{D_k}(i)$.

To derive the expression for updating $\bar{\boldsymbol{\omega}}_k(i)$, the following associated quantities are defined

$$\bar{\boldsymbol{\Phi}}_k(i) \triangleq \bar{\tilde{\boldsymbol{R}}}_k^{-1}(i) \tag{6.31}$$

where $\bar{\tilde{\boldsymbol{R}}}_k(i) = \sum_{l=1}^{i} \lambda^{i-l} \bar{\boldsymbol{x}}_k(i) \bar{\boldsymbol{x}}_k^H(i)$,

$$\bar{\boldsymbol{p}}_k(i) = \lambda \bar{\boldsymbol{p}}_k(i-1) + d_k^*(i) \boldsymbol{x}_k(i). \tag{6.32}$$

Then, equation (6.8) will be rewritten as

$$\begin{aligned} \bar{\boldsymbol{\omega}}_k(i) &= \bar{\tilde{\boldsymbol{R}}}_k^{-1}(i) \bar{\boldsymbol{p}}_k(i) \\ &= \bar{\boldsymbol{\Phi}}_k(i) \bar{\boldsymbol{p}}_k(i) \\ &= \lambda \bar{\boldsymbol{\Phi}}_k(i) \bar{\boldsymbol{p}}_k(i-1) + d_k^*(i) \bar{\boldsymbol{\Phi}}_k(i) \boldsymbol{x}_k(i) \\ &= \bar{\boldsymbol{\omega}}_k(i-1) + \bar{\boldsymbol{k}}_k(i) \Big[d_k^*(i) - \bar{\boldsymbol{x}}_k^H(i) \bar{\boldsymbol{\omega}}_k(i-1) \Big], \end{aligned} \tag{6.33}$$

where the $D \times 1$ Kalman gain vector is given by

$$\bar{\boldsymbol{k}}_k(i) = \frac{\lambda^{-1} \bar{\boldsymbol{\Phi}}_k(i-1) \bar{\boldsymbol{x}}_k(i)}{1 + \lambda^{-1} \bar{\boldsymbol{x}}_k^H(i) \bar{\boldsymbol{\Phi}}_k(i-1) \bar{\boldsymbol{x}}_k(i)}. \tag{6.34}$$

and the update for the matrix inverse $\bar{\boldsymbol{\Phi}}_k(i)$ employs the matrix inversion lemma [9]

$$\bar{\boldsymbol{\Phi}}_k(i) = \lambda^{-1} \bar{\boldsymbol{\Phi}}_k(i-1) - \lambda^{-1} \bar{\boldsymbol{k}}_k(i) \bar{\boldsymbol{x}}_k^H(i) \bar{\boldsymbol{\Phi}}_k(i-1). \tag{6.35}$$

Equations (6.31)–(6.35) constitute the key part of the proposed DRJIO-RLS algorithm for computing $\bar{\boldsymbol{\omega}}_k(i)$. The proposed DRJIO–RLS algorithm is detailed in Table 6.2.

6.4.3 Computational Complexity Analysis

Here, we evaluate the computational complexity of the proposed DRJIO–NLMS and DRJIO–RLS algorithms. The computational complexity of the proposed DRJIO–NLMS algorithm is $O(DM)$, while the proposed DRJIO–RLS algorithm has a complexity $O(M^2 + D^2)$. The distributed NLMS algorithm [2] has a complexity $O(M)$, while the complexity of the distributed RLS algorithm [20] is $O(M^2)$. For the Krylov Subspace NLMS [95] the complexity reaches $O(DM^2)$, while for the distributed principal subspace estimation algorithms [93], the complexity is $O(M^3)$. Thus, the proposed DRJIO–NLMS algorithm has a much lower computational complexity, and because $D \ll M$, it is as

Table 6.2: The DRJIO-RLS Algorithm

Initialize: $\bar{\boldsymbol{\omega}}_k(0)$=0

$\boldsymbol{P}_k(0) = \delta^{-1}\boldsymbol{I}_{M\times M}$, $\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(0) = \delta^{-1}\boldsymbol{I}_{D\times D}$,

$\bar{\boldsymbol{\Phi}}_k(0) = \delta^{-1}\boldsymbol{I}_{D\times D}$, $\delta =$ small positive constant and are same for all matrices

$\boldsymbol{S}_{D_k}^H(0) = [\boldsymbol{I}_D \;\; \boldsymbol{0}_{D,M-D}]^T, k = 1, 2, ..., N$

For each time instant $i = 1, 2, ..., I$

For each node $k = 1, 2, \ldots, N$

$$\boldsymbol{k}_k(i) = \frac{\lambda^{-1}\boldsymbol{P}_k(i-1)\boldsymbol{x}_k(i)}{1+\lambda^{-1}\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i-1)\boldsymbol{x}_k(i)}$$

$$\boldsymbol{t}_k(i) = \frac{\lambda^{-1}\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1)\bar{\boldsymbol{\omega}}_k(i-1)}{1+\lambda^{-1}\bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1)\bar{\boldsymbol{\omega}}_k(i-1)}$$

$$\boldsymbol{S}_{D_k}(i) = \boldsymbol{S}_{D_k}(i-1) + \boldsymbol{k}_k(i)\left[d_k^*(i)\boldsymbol{t}_k^H(i) - \boldsymbol{x}_k^H(i)\boldsymbol{S}_{D_k}(i-1)\right]$$

$$\boldsymbol{P}_k(i) = \lambda^{-1}\boldsymbol{P}_k(i-1) - \lambda^{-1}\boldsymbol{k}_k(i)\boldsymbol{x}_k^H(i)\boldsymbol{P}_k(i-1)$$

$$\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i) = \lambda^{-1}\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1) - \lambda^{-1}\boldsymbol{t}_k(i)\bar{\boldsymbol{\omega}}_k^H(i-1)\boldsymbol{Q}_{\bar{\boldsymbol{\omega}}_k}(i-1)$$

$$\bar{\boldsymbol{k}}_k(i) = \frac{\lambda^{-1}\bar{\boldsymbol{\Phi}}_k(i-1)\bar{\boldsymbol{x}}_k(i)}{1+\lambda^{-1}\bar{\boldsymbol{x}}_k^H(i)\bar{\boldsymbol{\Phi}}_k(i-1)\bar{\boldsymbol{x}}_k(i)}$$

$$\bar{\boldsymbol{\psi}}_k(i) = \bar{\boldsymbol{\omega}}_k(i-1) + \bar{\boldsymbol{k}}_k(i)\left[d_k^*(i) - \bar{\boldsymbol{x}}_k^H(i)\bar{\boldsymbol{\omega}}_k(i-1)\right]$$

$$\bar{\boldsymbol{\Phi}}_k(i) = \lambda^{-1}\bar{\boldsymbol{\Phi}}_k(i-1) - \lambda^{-1}\bar{\boldsymbol{k}}_k(i)\bar{\boldsymbol{x}}_k^H(i)\bar{\boldsymbol{\Phi}}_k(i-1)$$

end

For each node $k = 1, 2, \ldots, N$

$$\bar{\boldsymbol{\omega}}_k(i) = \sum_{l\in\mathcal{N}_k} c_{kl}\bar{\boldsymbol{\psi}}_l(i)$$

end

end

For each node $k = 1, 2, \ldots, N$

$\boldsymbol{\omega}_k(I) = \boldsymbol{S}_{D_k}(I)\bar{\boldsymbol{\omega}}_k(I)$

where $\boldsymbol{\omega}_k(I)$ is the final full–rank estimator.

end

simple as the distributed NLMS algorithm. An additional and very important aspect of distributed reduced–rank algorithms is that the dimensionality reduction results in a decrease in the number of transmitted parameters from M to D which corresponds to a less stringent bandwidth requirement. The details of computational complexity of the proposed and the existing algorithms, are shown in Table 6.3, where M is the length of the unknown parameter, D is the reduced dimension and $|\mathcal{N}_k|$ is the cardinality of $\mathcal{N}_k$.

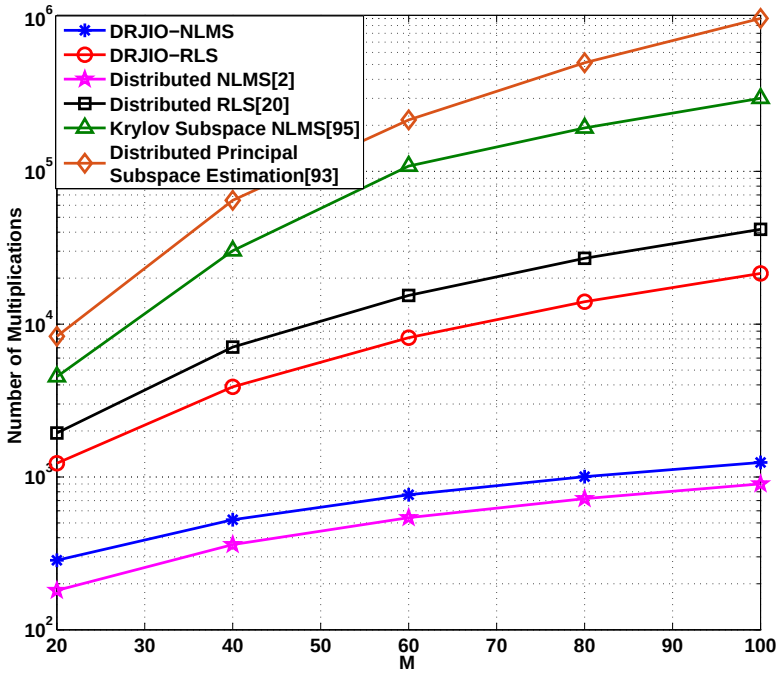

Figure 6.3: Complexity in terms of multiplications

To further illustrate the computational complexity for different algorithms, we present the main trends in terms of the number of multiplications for the proposed and existing algorithms in Fig. 6.3. We take node 14 as an example to calculate the computational complexity, where the parameters are $D = 5$ and $|\mathcal{N}_k| = 5$.

6.5 Simulation results

In this section, we investigate the performance of the proposed DRJIO–NLMS and DRJIO–RLS algorithms for distributed estimation in two scenarios: wireless sensor networks and smart grids.

Table 6.3: Computational Complexity of Different Algorithms per Node

Algorithm	Multiplications	Additions
DRJIO–NLMS	$2(D+1)M + (3+\|\mathcal{N}_k\|)D$ $+5$	$(2D+1)M + (2+\|\mathcal{N}_k\|)D$ -2
DRJIO–RLS	$2M^2 + (3+2D)M + 4D^2$ $+(9+\|\mathcal{N}_k\|)D$	$3M^2 + 2DM + 4D^2$ $+(2+\|\mathcal{N}_k\|)D$
Distributed NLMS [2]	$(4+\|\mathcal{N}_k\|)M + 1$	$(5+\|\mathcal{N}_k\|)M - 1$
Distributed RLS [20]	$4M^2 + (12+\|\mathcal{N}_k\|)M - 1$	$4M^2 + (16+\|\mathcal{N}_k\|)M + 1$
Krylov Subspace NLMS [95]	$6DM^2 + 4M + (5+\|\mathcal{N}_k\|)D$	$6DM^2 + 2M + (2+\|\mathcal{N}_k\|)D$
Distributed principal subspace estimation [93]	$M^3 + 2(D+2)M$ $+(3+\|\mathcal{N}_k\|)D + 4$	$M^3 + (D+1)M$ $+(2+\|\mathcal{N}_k\|)D - 1$

6.5.1 Wireless Sensor Networks

In this subsection, we compare our proposed DRJIO–NLMS algorithm and DRJIO–RLS algorithm with the distributed NLMS algorithm (normalized version of [2]), distributed RLS algorithm [20], Krylov subspace NLMS [95] and distributed principal subspace estimation [93], based on their MSE performance. With the network topology structure outlined in Fig. 6.1 with $N = 20$ nodes, we consider numerical simulations under three scenarios:

- Full–rank system with M=20
- Sparse system with M=20 (D valid coefficients and $M - D$ zeros coefficients)
- Full–rank system with M=60

The input signal is generated as $\boldsymbol{x}_k(i) = [x_k(i) \quad x_k(i-1) \quad ... \quad x_k(i-M+1)]$ and $x_k(i) = u_k(i) + \alpha_k x_k(i-1)$, where α_k is a correlation coefficient and $u_k(i)$ is a white noise process with variance $\sigma^2_{u,k} = 1 - |\alpha_k|^2$, to ensure the variance of $\boldsymbol{x}_k(i)$ is $\sigma^2_{x,k} = 1$. The noise samples are modeled as complex Gaussian noise with variance of $\sigma^2_{n,k} = 0.001$. We assume that the network has error–free transmission between linked nodes.

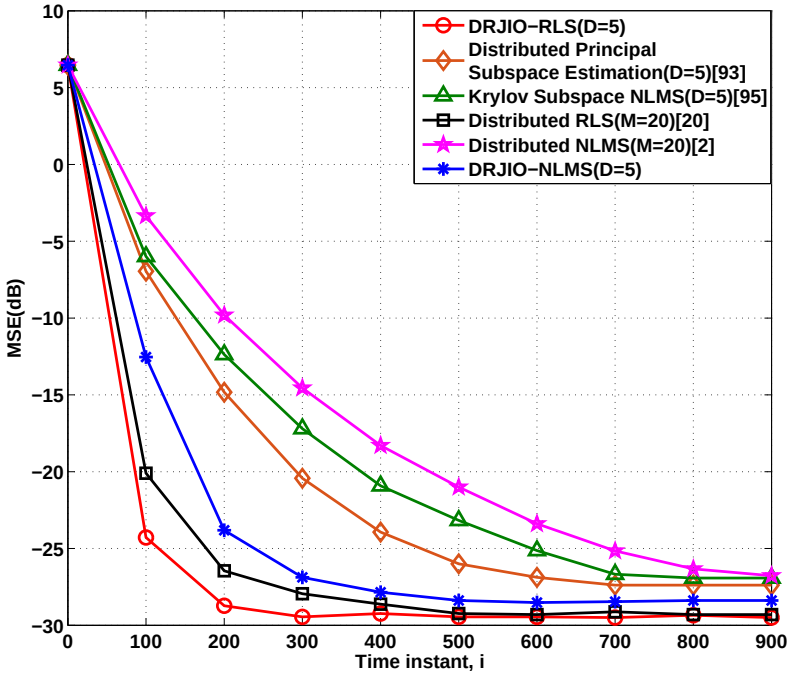

Figure 6.4: Full–rank system with M=20

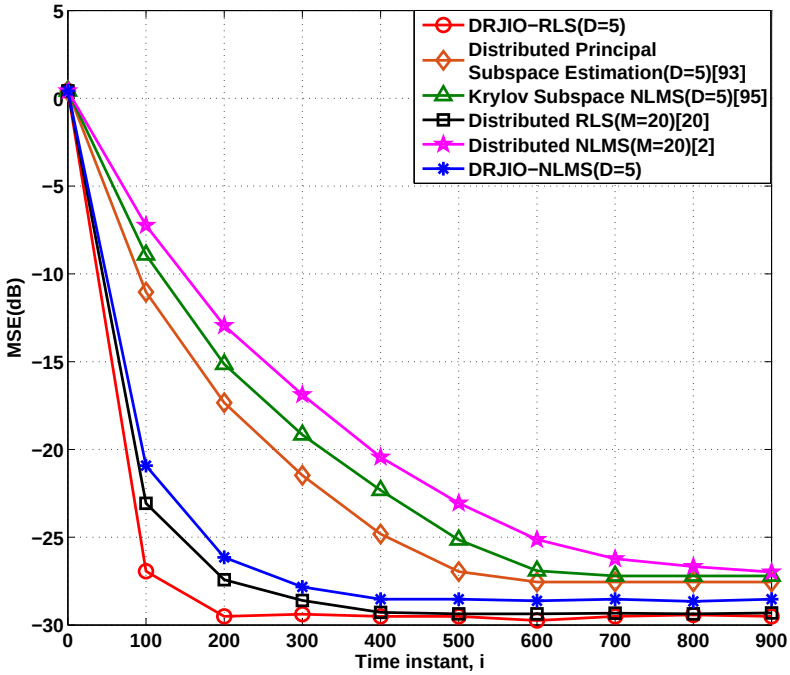

Figure 6.5: Sparse system with M=20

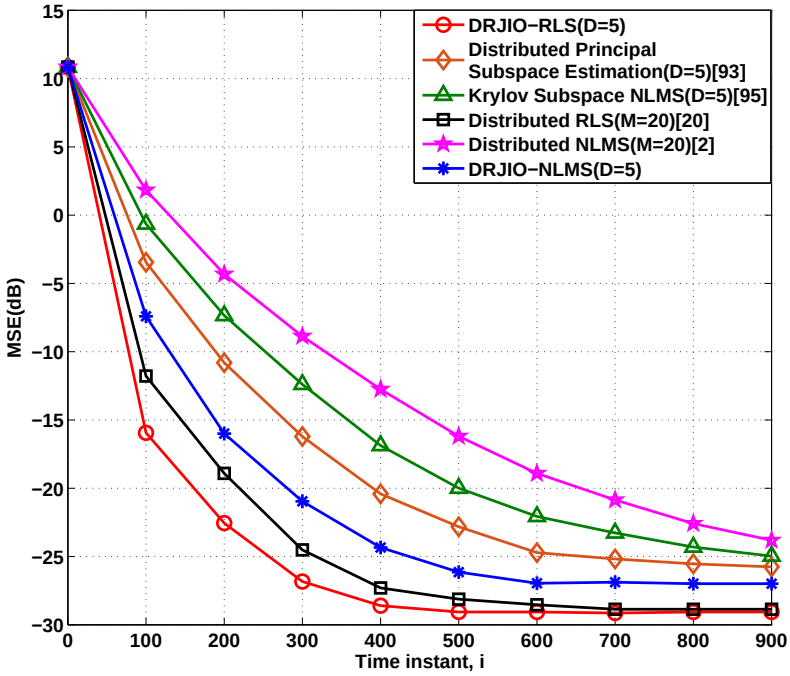

Figure 6.6: Full–rank system with M=60

The step size μ_0 for the distributed NLMS algorithm, Krylov subspace NLMS, distributed principal subspace estimation and DRJIO–NLMS is set to 0.15 and η_0 is set to 0.5. For the distributed RLS algorithm and DRJIO–RLS algorithm, the forgetting factor λ is equal to 0.99 and δ is set to 0.11. In Fig. 6.4, we compare the proposed DRJIO–NLMS, DRJIO–RLS with the existing strategies using the full–rank system with M=20 and D=5. The dimensionality reduction matrix $\boldsymbol{S}_{D_k}(0)$ is initialized as $[\boldsymbol{I}_D \;\; \boldsymbol{0}_{D,M-D}]^T$. We observe that the proposed DRJIO–RLS algorithm has the best performance when compared with other algorithms, while the proposed DRJIO–NLMS algorithm also has a better performance, which is very close to the distributed RLS algorithm. However, its complexity is an order of magnitude lower than the distributed RLS algorithm and DRJIO–RLS algorithm.

In a sparse system scenario with M=20, the proposed DRJIO–RLS and DRJIO–NLMS algorithms still have excellent performance as shown in Fig. 6.5. Specifically, the proposed DRJIO–NLMS algorithm performs very close to the distributed RLS algorithm and outperforms the other analyzed algorithms. When the full–rank system M increases to 60, Fig. 6.6 illustrates that, the proposed DRJIO–RLS algorithm still has the best per-

formance, while DRJIO–NLMS algorithm also shows a fast convergence rate, which is comparable to that of the distributed RLS algorithm. For the distributed NLMS, Krylov subspace NLMS and distributed principal subspace estimation algorithms, their convergence speed is much lower.

In the last example on wireless sensor networks, we compare the performance between the proposed DRJIO–NLMS and DCE scheme in Chapter 5, under different sparsity level scenarios. The step size for both algorithms are set to 0.3 and the η_0 for DRJIO–NLMS is set to 0.5. The length of the unknown parameter $\boldsymbol{\omega}_0$ is 20 and $D = 10$. For the first scenario, the number of non–zero coefficients in the unknown parameter is 3 and for the second scenario, the number of non–zero coefficients is set to 10. The comparison results are shown in Fig. 6.7 and 6.8. It is clear that in a very sparse system, the proposed DCE scheme performs better than the DRJIO–NLMS algorithm. With the decrease of system sparse level, the proposed DRJIO–NLMS algorithm performs better than the DCE scheme.

6.5.2 Smart Grids

In order to test the proposed algorithms in a possible smart grid scenario, we consider the Hierarchical IEEE 14–bus system which has been proposed in [104], where 14 is the number of substations. At every time instant i, each bus $k, k = 1, 2, \ldots, 14$, takes a scalar measurement $d_k(i)$ according to

$$d_k(i) = X_k\big(\boldsymbol{\omega}_0(i)\big) + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{6.36}$$

where $\boldsymbol{\omega}_0(i)$ is the state vector of the entire interconnected system, $X_k(\boldsymbol{\omega}_0(i))$ is a non-linear measurement function of bus k. The quantity $n_k(i)$ is the measurement error with mean equal to zero and which corresponds to bus k.

We focus on the linearized DC state estimation problem. We assume that each bus connects and measures three users' state, as a result, for the IEEE–14 bus system, there will be 42 users in the system. The state vector $\boldsymbol{\omega}_0(i)$ is taken as the voltage phase angle vector $\boldsymbol{\omega}_0$ for all users. Initially, each bus only knows the voltage phase angle of the three users connected to it. With the help of distributed estimation algorithms, each bus

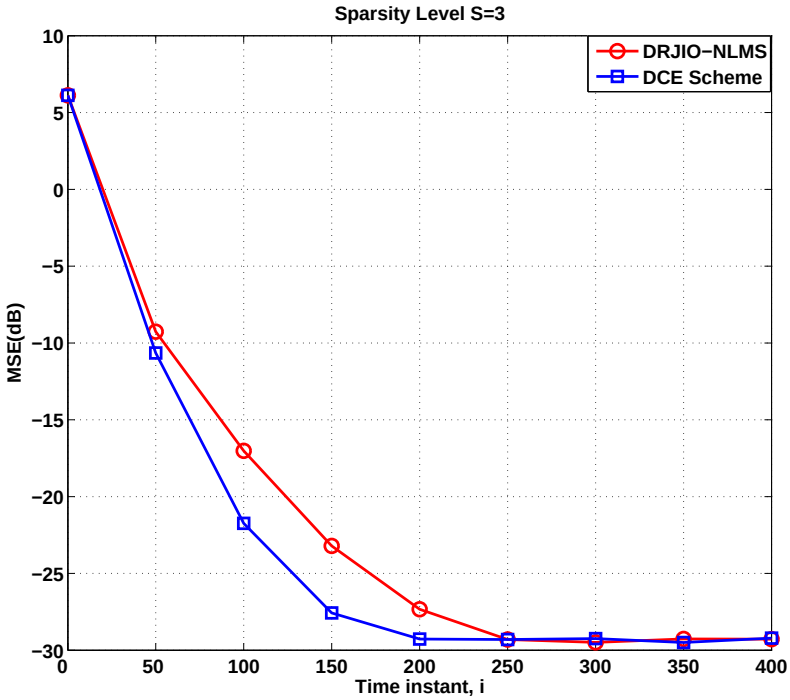

Figure 6.7: DRJIO–NLMS vs DCE scheme with sparsity level S=3

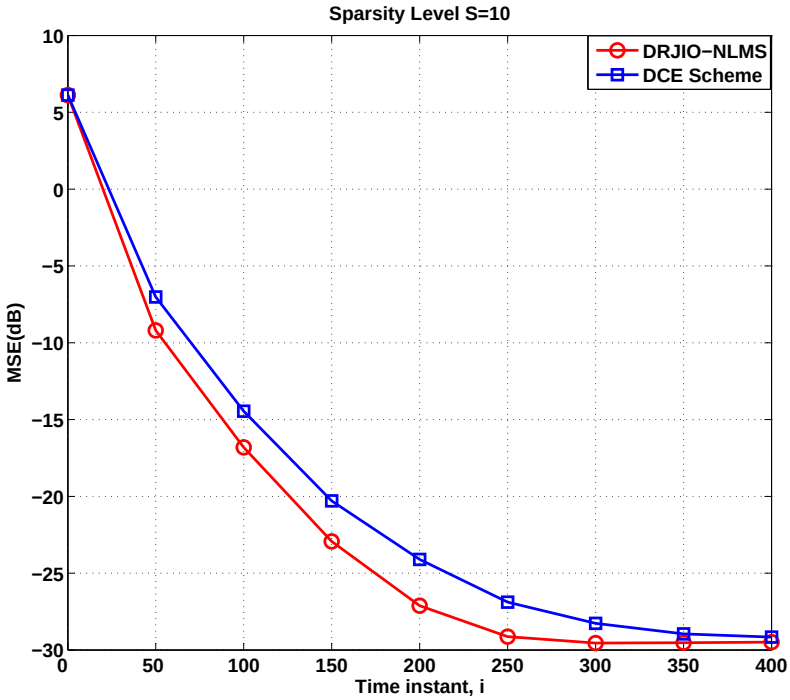

Figure 6.8: DRJIO–NLMS vs DCE scheme with sparsity level S=10

is supposed to estimate the state of the voltage phase angles for all users in the system. Therefore, the nonlinear measurement model for state estimation (6.36) is approximated by

$$d_k(i) = \boldsymbol{x}_k^H(i)\boldsymbol{\omega}_0 + n_k(i), \quad k = 1, 2, \ldots, 14, \tag{6.37}$$

where $\boldsymbol{x}_k(i)$ is the measurement Jacobian vector for bus k. Then, the aim of the distributed estimation algorithm is to compute an estimate of $\boldsymbol{\omega}_0$, which can minimize the cost function given by

$$J_{\boldsymbol{\omega}_k(i)}(\boldsymbol{\omega}_k(i)) = \mathbb{E}|d_k(i) - \boldsymbol{x}_k^H(i)\boldsymbol{\omega}_k(i)|^2. \tag{6.38}$$

and the global network cost function is described by

$$J_\omega(\boldsymbol{\omega}) = \sum_{k=1}^{N} \mathbb{E}|d_k(i) - \boldsymbol{x}_k^H(i)\boldsymbol{\omega}|^2. \tag{6.39}$$

We compare the proposed algorithms with the $\mathcal{M}$–$\mathcal{CSE}$ algorithm [4], the distributed RLS algorithm [20], the distributed NLMS algorithm (normalized version of [2]) and distributed principal subspace estimation [93] in terms of MSE performance. The MSE comparison is used to determine the accuracy of the algorithms and the rate of convergence. We define the Hierarchical IEEE–14 bus system as in Fig. 6.9.

All buses are corrupted by additive white Gaussian noise with variance $\sigma_{n,k}^2 = 0.001$. The step size for the distributed NLMS [2] and the proposed DRJIO–NLMS algorithms is $\mu_0 = 0.15$ and η_0 is set to 0.5. The parameter vector $\boldsymbol{\omega}_0$ is set to an all–one vector with size 42×1. For the distributed RLS, DRJIO–RLS algorithms the forgetting factor λ is set to 0.99 and δ is equal to 0.11. The reduced dimension D is set to 10 for both DRJIO–RLS and DRJIO–NLMS algorithm. The results are averaged over 100 independent runs. We simulate the proposed algorithms for smart grids under a static scenario.

From Fig. 6.10, it can be seen that the proposed DRJIO–RLS algorithm has the best performance, and significantly outperforms the distributed NLMS [2] and the $\mathcal{M}$–$\mathcal{CSE}$ [4] algorithms. The DRJIO–NLMS is slightly worse than distributed RLS algorithm [20], but better than the distributed NLMS and $\mathcal{M}$–$\mathcal{CSE}$ algorithms. In addition, the proposed DRJIO–NLMS and DRJIO–RLS algorithms can compress the data to be transmitted from each node from M to D, resulting in reduced bandwidth requirements. These algorithms are also important tools for dealing with large sets of data which exhibit some form of redundancy, sparsity and are compressible.

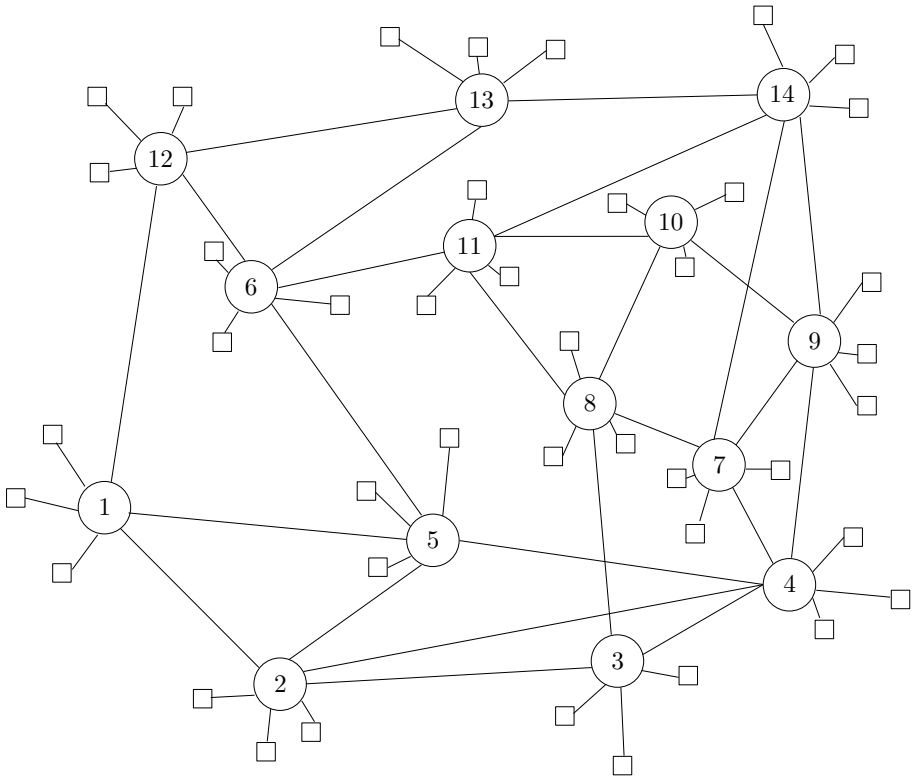

Figure 6.9: Hierarchical IEEE 14–bus system

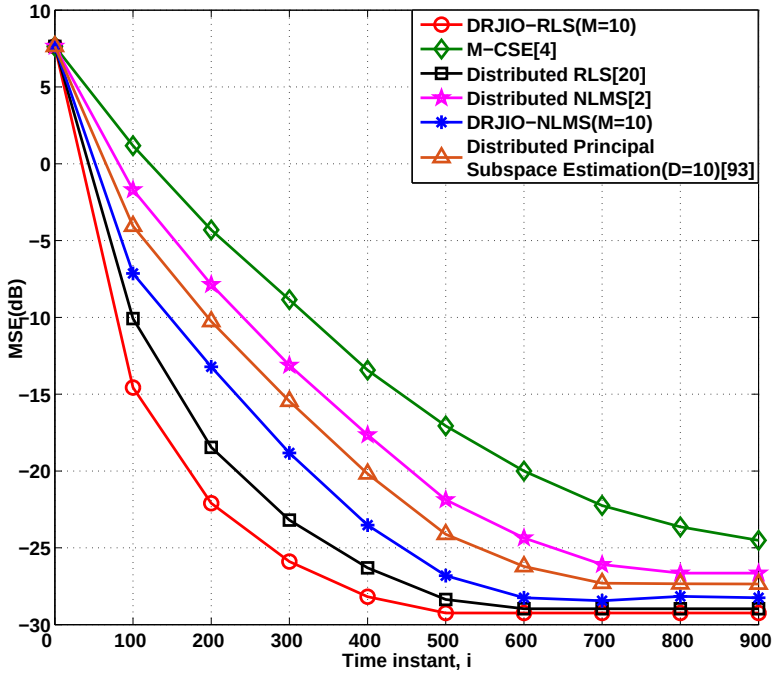

Figure 6.10: MSE performance for smart grids

6.6 Summary

In this chapter, we have proposed a novel distributed reduced–rank scheme along with efficient algorithms for distributed estimation in wireless sensor networks and smart grids. Simulation results have shown that the proposed DRJIO–RLS has the best performance, while DRJIO–NLMS algorithm has a better performance and lower cost than existing algorithms apart from the distributed RLS algorithm, in all the three scenarios considered. We have also compared the proposed algorithms with the DCE scheme, which was presented in chapter 5, for systems with different levels of sparsity. Furthermore, the proposed scheme requires the transmission of only D parameters instead of M, resulting in higher bandwidth efficiency than standard schemes.

Chapter 7

Conclusions and Future Work

Contents

7.1 Summary of the Work 135
7.2 Future Work 137

7.1 Summary of the Work

In this thesis, a number of innovative distributed cooperative strategies for dealing with exchange of information, node failures and compression of data have been considered for wireless sensor networks, spectrum estimation and smart grids, which require low complexity and high performance algorithms. The proposed distributed algorithms have been studied in statistical inference problems and found to be highly effective and to outperform previously reported algorithms in the literature in several applications.

In Chapter 3, distributed adaptive algorithms based on the conjugate CG method for distributed networks have been presented. Both incremental and diffusion adaptive solutions are considered. In particular, the IDCCG/IDMCG and DDCCG/DDMCG algorithms have been proposed for applications to parameter and spectrum estimation. The distributed CCG and MCG algorithms provide an improved performance in terms of MSE as compared with LMS–based algorithms and a performance that is close to RLS algorithms. The design of preconditioners for CG algorithms, which have the ability to im-

prove the performance of the proposed CG algorithms is also presented in this chapter. The resulting algorithms are distributed, cooperative and able to respond in real time to changes in the environment. Simulation results have proved the advantages of the proposed IDCCG/IDMCG and DDCCG/DDMCG algorithms in different applications. The distributed CG algorithms outperform LMS algorithms and have a close performance to RLS algorithms without the numerical problems of the latter. Designers could implement the distributed CG algorithms in hardware and deploy sensor networks based on these tools.

In Chapter 4, adaptive link selection algorithms for distributed estimation and their application to wireless sensor networks and smart grids have been investigated. Specifically, based on the LMS/RLS strategies, exhaustive search-based LMS/RLS link selection algorithms and sparsity-inspired LMS/RLS link selection algorithms that can exploit the topology of networks with poor-quality links have been considered. The proposed link selection algorithms were then analyzed in terms of their stability, steady-state and tracking performance, and computational complexity. We have compared the proposed algorithms with existing methods. In comparison with existing centralized or distributed estimation strategies, more accurate estimates and faster convergence speed can be obtained for the proposed algorithms and the network is equipped with the ability of link selection that can circumvent link failures and improve the estimation performance. We have also devised analytical expressions to predict their MSE steady–state performance and tracking behavior. Simulation experiments have been conducted to verify the analytical results and illustrate that the proposed algorithms significantly outperform the existing strategies, in both static and time–varying scenarios, in examples of wireless sensor networks and smart grids. Furthermore, the proposed link selection algorithms could be used in any distributed scenario where the topology is not optimized in order to further improve the performance.

In Chapter 5, a novel distributed compressed estimation scheme, namely DCE scheme, has been introduced for sparse signals and systems based on compressive sensing techniques. The proposed scheme consists of compression and decompression modules inspired by compressive sensing to perform distributed compressed estimation. In the DCE scheme, the estimation procedure has been performed in a compressed dimension. A design procedure has also been presented and an algorithm developed to optimize mea-

surement matrices, which can further improve the performance of the proposed distributed compressed estimation scheme. The simulation results for a WSN application have show that the DCE scheme outperforms existing strategies in terms of convergence rate, reduced bandwidth and MSE performance. Designers could implement the DCE scheme in hardware for scenarios with sparse signals and deploy sensor networks for detection and estimation based on this scheme.

In Chapter 6, the challenge of estimating large dimension unknown parameter vectors in wireless sensor networks and smart grids that requires large communication bandwidth is addressed. A novel distributed reduced-rank scheme and adaptive algorithms have been proposed for distributed estimation in wireless sensor networks and smart grids. In particular, we have developed a novel dimensionality reduction scheme and adaptive algorithms for performing distributed dimensionality reduction and computing low–rank approximations of unknown parameter vectors. The proposed scheme is based on a transformation that performs dimensionality reduction at each agent of the network followed by a reduced–dimension parameter vector. The DRJIO–NLMS and DRJIO–RLS have been developed to achieve significantly reduced communication overhead and improved performance. Simulation results illustrate the advantages of the proposed strategy in terms of convergence rate and MSE performance. In addition, the proposed DRJIO–NLMS and DRJIO–RLS algorithms could be used in any distributed scenario for large sets of data that exhibit some level of sparsity or redundancy in order to further improve the performance.

7.2 Future Work

Many of the proposed schemes and algorithms detailed in this thesis have potential to be applied to scenarios, systems and techniques outside the scope of this thesis, and there is further work and analysis that could be considered to extend the work that has been covered.

The proposed distributed CG based algorithms in chapter 3 can be extended to a sparsity–aware version and a cooperation with CETUC, PUC–Rio will be carried out

on the topic of ”Sparsity–Aware Distributed Conjugate Gradient Algorithms for Spectrum Sensing”. This could be particularly useful in spectrum sensing problems where the spectrum content can be modelled as a sparse parameter vector and exploited via sparsity–aware distributed CG algorithms.

New distributed network cooperative protocols can also be investigated and compared with current protocols proposed in the thesis. Multi–task distributed processing, where the task is to estimate multiple parameter vectors, can also be considered to deal with parameter estimation in big data problems and heterogeneous networks.

In chapter 4, the proposed link selection strategies softly change the network topology in wireless sensor networks and smart grids by only employing a selected subset of links, which correspond to the effective topology used for a desired task. Further research on physical dynamic topology adaptation strategies can be carried out as future work.

The algorithms and systems described in this thesis have not considered the channel state between linked nodes. In order to broaden the application of the algorithms in more realistic scenarios, the flat fading or fast fading channel design may be considered in the future.

Glossary

AP	**A**ffine **P**rojection
ATC	**A**dapte **T**hen **C**ombin
CCG	**C**onventional **C**onjugate **G**radient
CG	**C**onjugate **G**gradient
CoSaMP	**Co**mpressive **Sa**mpling **M**atching **P**ursuit
CS	**C**ompressive **S**ensing
CTA	**C**ombine **T**hen **A**dapt
DC	**D**irect **C**urrent
DCE	**D**istributed **C**ompressed **E**stimation
DCT	**D**iscrete **C**osine **T**ransform
DDCCG	**D**iffusion **D**istributed **C**onventional **C**onjugate **G**gradient
DDMCG	**D**iffusion **D**istributed **M**odified **C**onjugate **G**gradient
DFT	**D**iscrete **F**ourier **T**ransform
dNLMS	**d**istributed **N**ormalized **L**east **M**ean **S**quare
DRJIO–NLMS	**D**istributed **R**educed-rank **J**oint **I**terative **O**ptimization–**N**ormalized **L**east **M**ean **S**quare
DRJIO–RLS	**D**istributed **R**educed-rank **J**oint **I**terative **O**ptimization–**R**ecursive **L**east **S**quare
ES–LMS	**E**xhaustive **S**earch–based **L**east **M**ean **S**quare
EMSE	**E**xcess **M**ean **S**quare **E**rror
ES–RLS	**E**xhaustive **S**earch–based **R**ecursive **L**east **S**quare
FIR	**F**inite **I**mpulse **R**esponse
IDCCG	**I**ncremental **D**istributed **C**onventional **C**onjugate **G**gradient
IDMCG	**I**ncremental **D**istributed **M**odified **C**onjugate **G**gradient

iid	**i**ndependent and **i**dentically **d**istributed
KLT	**K**ahunen–**L**oève **T**ransform
LMS	**L**east **M**ean **S**quare
MCG	**M**odified **C**onjugate **G**gradient
MIMO	**M**ulti **I**nput **M**ulti **O**utput
MMSE	**M**inimum **M**ean **S**quare **E**rror
MSE	**M**ean **S**quare **E**rror
MSD	**M**ean **S**quare **D**eviation
OMP	**O**rthogonal **M**atching **P**ursuit
PSD	**P**ower **S**pectral **D**ensity
RIP	**R**estricted **I**sometry **P**roperty
RLS	**R**ecursive **L**east **S**quare
RZA–LMS	**R**eweighted **Z**ero **A**ttracting–**L**east **M**ean **S**quare
SI–LMS	**S**parsity–**I**nspired **L**east **M**ean **S**quare
SI–RLS	**S**parsity–**I**nspired **R**ecursive **L**east **S**quare
StOMP	**St**agewise **O**rthogonal **M**atching **P**ursuit
WSNs	**W**ireless **S**ensor **N**etworks
ZA–LMS	**Z**ero **A**ttracting–**L**east **M**ean **S**quare

Bibliography

[1] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 223–229, August 2007.

[2] ——, "Diffusion least–mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, July 2008.

[3] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for system identification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, April 2009, pp. 3125–3128.

[4] L. Xie, D. H. Choi, S. Kar, and H. V. Poor, "Fully distributed state estimation for wide-area monitoring systems," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1154–1169, September 2012.

[5] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E90-A, no. 8, pp. 1504–1510, August 2007.

[6] O. Axelsson, *Iterative Solution Methods*. New York, NY, USA: Cambridge University Press, 1995.

[7] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[8] P. S. Chang and J. A. N. Willson, "Analysis of conjugate gradient algorithms for adaptive filtering," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 409–418, February 2000.

[9] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[10] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: John Wiley&Sons, 2003.

[11] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theor.*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[12] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal. Proc. Mag.*, vol. 25, no. 2, pp. 21–30, March 2008.

[13] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank processing based on joint and iterative interpolation, decimation and filtering," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2503–2514, July 2009.

[14] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning.*, vol. 1, no. 4-5, pp. 311–801, July 2014.

[15] ——, "Diffusion adaptation over networks," *E-Reference Signal Processing*, R. Chellapa and S. Theodoridis, editors, Elsevier, 2013.

[16] L. Li and J. A. Chambers, "Distributed adaptive estimation based on the APA algorithm over diffusion networks with changing topology," in *Proc. IEEE/SP 15th Workshop on Statistical Signal Processing*, Cardiff, Wales, September 2009, pp. 757–760.

[17] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, pp. 1035–1048, March 2010.

[18] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed Recursive Least-Squares for Consensus-Based In-Network Adaptive Estimation," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4583–4588, November 2009.

[19] A. Bertrand and M. Moonen, "Distributed adaptive node–specific signal estimation in fully connected sensor networks–part II: Simultaneous and asynchronous node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5292–5306, October 2010.

[20] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[21] F. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed kalman filtering and smoothing," *IEEE Trans. Autom. Control*, vol. 55, no. 9, pp. 2069–2084, September 2010.

[22] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, March 2010.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM JOURNAL ON SCIENTIFIC COMPUTING*, vol. 20, pp. 33–61, 1998.

[24] Y. V. Zakharov, T. C. Tozer, and J. F. Adlard, "Polynomial spline–approximation of clarke's model," *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1198–1208, May 2004.

[25] P. D. Lorenzo, S. Barbarossa, and A. H. Sayed, "Distributed spectrum estimation for small cell networks based on sparse diffusion adaptation," *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1261–1265, December 2013.

[26] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus–based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, June 2009.

[27] A. Bose, "Smart transmission grid applications and their supporting infrastructure," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 11–19, June 2010.

[28] X. Zhao and A. H. Sayed, "Single-link diffusion strategies over adaptive networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, March 2012, pp. 3749–3752.

[29] S. Xu, R. C. de Lamare, and H. V. Poor, "Dynamic topology adaptation for distributed estimation in smart grids," in *Proc. IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, December 2013, pp. 420–423.

[30] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, September 2004.

[31] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control.*, vol. 49, pp. 1520–1533, September 2004.

[32] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[33] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1996.

[34] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, July 2007.

[35] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theor.*, vol. 52, no. 2, pp. 489–509, February 2006.

[36] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theor.*, vol. 51, no. 12, pp. 4203–4215, December 2005.

[37] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, December 1993.

[38] J. A. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inform. Theory*, vol. 53, no. 12, pp. 4655–4666, December 2007.

[39] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. Inf. Theor.*, vol. 58, no. 2, pp. 1094–1121, February 2012.

[40] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmonic Anal.*, vol. 26, no. 3, pp. 301–321, May 2009.

[41] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comp. Harmonic Anal.*, vol. 27, no. 3, pp. 265–274, November 2009.

[42] R. G. Baraniuk, M. Davenport, R. DeVore, and M. B. Wakin, "A simple proof of the restricted isometry principle for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, Springer New York, 2008.

[43] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, Pacic Grove, CA, November 1993, pp. 40–44 vol.1.

[44] P. D. Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *IEEE Trans. Signal Process.*, vol. 61, no. 6, pp. 1419–1433, March 2013.

[45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[46] E. J. Candes, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, pp. 877–905, 2007.

[47] S. Xu, R. C. de Lamare, and H. V. Poor, "Adaptive link selection strategies for distributed estimation in diffusion wireless networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2013, pp. 5402–5405.

[48] A. H. Sayed and C. G. Lopes, "Distributed recursive least-squares strategies over adaptive networks," in *Fortieth Asilomar Conference on Signals, Systems and Computers*, October 2006, pp. 233–237.

[49] S. Xu, R. C. de Lamare, and H. V. Poor, "Distributed compressed estimation based on compressive sensing," *IEEE Signal Processing Letters*, vol. 22, no. 9, pp. 1311–1315, September 2015.

[50] M. S. E. Abadi and Z. Saffari, "Distributed estimation over an adaptive diffusion network based on the family of affine projection algorithms," in *Sixth International Symposium on Telecommunications*, November 2012, pp. 607–611.

[51] L. Wang and R. C. de Lamare, "Constrained adaptive filtering algorithms based on conjugate gradient techniques for beamforming," *IET Signal Processing*, vol. 4, no. 6, pp. 686–697, December 2010.

[52] P. S. Chang and J. A. N. Willson, "Adaptive filtering using modified conjugate gradient," in *Proceedings., Proceedings of the 38th Midwest Symposium on Circuits and Systems*, vol. 1, August 1995, pp. 243–246.

[53] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester, U.K.: Wiley, 1987.

[54] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 244–256, 1978.

[55] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5107–5124, October 2012.

[56] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[57] S. C. Eisenstat, "Efficient implementation of a class of preconditioned conjugate gradient methods," *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 1, pp. 1–4, 1981.

[58] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 23, no. 2, pp. 517–541, 2001.

[59] O. Axelsson and G. Lindskog, "On the rate of convergence of the preconditioned conjugate gradient method," *Numerische Mathematik*, vol. 48, no. 5, pp. 499–523, 1986.

[60] D. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim*, vol. 7, no. 4, pp. 913–926, 1997.

[61] A. Nedic and D. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim*, vol. 12, no. 1, pp. 109–138, 2001.

[62] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Sel. Areas Commun*, vol. 23, no. 4, pp. 798–808, April 2005.

[63] P. D. Lorenzo, S. Barbarossa, and A. H. Sayed, "Sparse diffusion LMS for distributed adaptive estimation," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Kyoto, Japan, March 2012, pp. 3281–3284.

[64] C. G. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, March 2008, pp. 3285–3288.

[65] B. Fadlallah and J. Principe, "Diffusion Least-Mean squares over adaptive networks with dynamic topologies," in *Proc. IEEE International Joint Conference on Neural Networks*, Dallas, TX, USA, August 2013, pp. 1–6.

[66] T. Wimalajeewa and S. K. Jayaweera, "Distributed node selection for sequential estimation over noisy communication channels," *IEEE Trans. Wirel. Commun.*, vol. 9, no. 7, pp. 2290–2301, July 2010.

[67] R. C. de Lamare and P. S. R. Diniz, "Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression," *IEEE Trans. Vehi. Techn.*, vol. 58, no. 2, pp. 644–654, February 2009.

[68] L. Guo and Y. F. Huang, "Frequency-domain set-membership filtering and its applications," *IEEE Trans. Signal Process.*, vol. 55, no. 4, pp. 1326–1338, April 2007.

[69] R. Meng, R. C. de Lamare, and V. H. Nascimento, "Sparsity-aware affine projection adaptive algorithms for system identification," in *Proc. Sensor Signal Processing for Defence*, London, UK, September 2011, pp. 1–5.

[70] Y. Chen, Y. Gu, and A. Hero, "Regularized least-mean-square algorithms," 2009 [Online]. Available: http://arxiv.org/abs/1012.5066.

[71] S. Xu and R. C. de Lamare, "Distributed conjugate gradient strategies for distributed estimation over sensor networks," in *Proc. Sensor Signal Processing for Defence*, London, UK, September 2012, pp. 1–5.

[72] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2000.

[73] R. C. de Lamare and P. S. R. Diniz, "Blind adaptive interference suppression based on set-membership constrained constant-modulus algorithms with dynamic bounds," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1288–1301, March 2013.

[74] Y. Cai and R. C. de Lamare, "Low–complexity variable step-size mechanism for code-constrained constant modulus stochastic gradient algorithms applied to cdma interference suppression," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 313–323, January 2009.

[75] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

[76] E. Eweda, "Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels," *IEEE Trans. Signal Process.*, vol. 42, pp. 2937–2944, November 1994.

[77] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: Where RLS meets the ℓ_1–norm," *IEEE Trans. Signal Process.*, vol. 58, no. 7, pp. 3436–3447, July 2010.

[78] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, "A sparsity promoting adaptive algorithm for distributed learning," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5412–5425, October 2012.

[79] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, October 2010.

[80] Z. Liu, Y. Liu, and C. Li, "Distributed sparse recursive least-squares over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1386–1395, March 2014.

[81] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5308–5323, October 2014.

[82] J. Romberg, "Imaging via compressive sampling," *IEEE Signal. Proc. Mag.*, vol. 25, no. 2, pp. 14–20, March 2008.

[83] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proc. The Fifth International Conference on Information Processing in Sensor Networks*, Nashville, TN, April 2006, pp. 134–142.

[84] Y. Yao, A. P. Petropulu, and H. V. Poor, "MIMO radar using compressive sampling," *IEEE J. Sel. Top. Sign. Proces.*, vol. 4, no. 1, pp. 146–163, February 2010.

[85] C. Wei, M. Rodrigues, and I. Wassell, "Distributed compressive sensing reconstruction via common support discovery," in *IEEE International Conference on Communications*, Kyoto, Japan, June 2011, pp. 1–5.

[86] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, "Distributed compressive sensing," in *ECE Department Tech. Report TREE-0612*, Rice University, Houston, TX, November 2006.

[87] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, compression, and recovery for WSNs: Sparse signal modeling and monitoring framework," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3447–3461, October 2012.

[88] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[89] Y. Yao, A. Petropulu, and H. V. Poor, "Measurement matrix design for compressive sensing–based MIMO radar," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5338–5352, November 2011.

[90] E. Msechu, S. Roumeliotis, A. Ribeiro, and G. Giannakis, "Decentralized quantized Kalman filtering with scalable communication cost," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3727–3741, October 2008.

[91] J. J. Xiao, A. Ribeiro, Z. Q. Luo, and G. Giannakis, "Decentralized quantized Kalman filtering with scalable communication cost," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, July 2006.

[92] S. Pereira and A. Pages-Zamora, "Distributed consensus in wireless sensor networks with quantized information exchange," in *Proc. IEEE 9th Workshop on Signal Processing Advances in Wireless Communications*, July 2008, pp. 241–245.

[93] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 725–738, August 2011.

[94] M. O. Sayin and S. Kozat, "Single bit and reduced dimension diffusion strategies over distributed networks," *IEEE Signal Process. Lett.*, vol. 20, no. 10, pp. 976–979, October 2013.

[95] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Trading off complexity with communication costs in distributed adaptive learning via Krylov subspaces for dimensionality reduction," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 257–273, April 2013.

[96] R. C. de Lamare and R. Sampaio-Neto, "Reduced–rank adaptive filtering based on joint iterative optimization of adaptive filters," *IEEE Signal Process. Lett.*, vol. 14, no. 12, pp. 980–983, December 2007.

[97] Y. Cai and R. C. de Lamare, "Adaptive linear minimum BER reduced–rank interference suppression algorithms based on joint and iterative optimization of filters," *IEEE Commun. Lett.*, vol. 17, no. 4, pp. 633–636, April 2013.

[98] P. Clarke and R. C. de Lamare, "Low–complexity reduced-rank linear interference suppression based on set-membership joint iterative optimization for DS–CDMA systems," *IEEE Trans. Veh. Technol.*, vol. 60, no. 9, pp. 4324–4337, November 2011.

[99] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank equalization algorithms based on alternating optimization design techniques for MIMO systems," *IEEE Trans. Vehi. Techn.*, vol. 60, no. 6, pp. 2482–2494, July 2011.

[100] ——, "Reduced-rank space–time adaptive interference suppression with joint iterative least squares algorithms for spread-spectrum systems," *IEEE Trans. Vehi. Techn.*, vol. 59, no. 3, pp. 1217–1228, March 2010.

[101] S. Nuan, W. U. Alokozai, R. C. de Lamare, and M. Haardt, "Adaptive widely linear reduced–rank beamforming based on joint iterative optimization," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 265–269, March 2014.

[102] L. Wang and R. C. de Lamare, "Low–complexity constrained adaptive reduced-rank beamforming algorithms," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 4, pp. 2114–2128, October 2013.

[103] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 386–398, January 2011.

[104] H. Ma, Y. Yang, Y. Chen, and K. J. R. Liu, "Distributed state estimation in smart grid with communication constraints," in *Asia–Pacific Signal Information Processing Association Annual Summit and Conference*, December 2012, pp. 1–4.